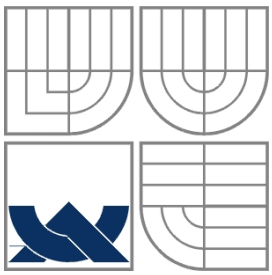


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

PLÁNOVÁNÍ CESTY MOBILNÍHO ROBOTU POMOCÍ MRAVENČÍCH ALGORITMŮ

MOBILE ROBOT PATH PLANNING BY MEANS OF ANT ALGORITHMS

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE **Bc. Václav Sedlák**
AUTHOR

VEDOUCÍ PRÁCE **RNDr. Jiří Dvořák, CSc.**
SUPERVISOR

BRNO 2011

ZADÁNÍ ZÁVĚREČNÉ PRÁCE

ABSTRAKT

Tato práce se zabývá plánováním cesty robota při použití algoritmů pro optimalizaci pomocí mravenčích kolonií. Teoretická část práce uvádí základní pojmy z problematiky plánování cesty robota. Dále se teoretická část věnuje mravenčím algoritmům coby nástrojům pro optimalizaci a plánování cesty robota. Praktická část práce se zabývá návrhem a implementací mravenčích algoritmů pro plánování cesty robota v jazyce Java.

ABSTRACT

This thesis deals with robot path planning by means of ant colony optimization algorithms. The theoretical part of this thesis introduces basics of path planning problematics. The theoretical part either deals with ant algorithms as optimization and path planning tools. The practical part deals with design and implementation of path planning by means of ant algorithms in Java language.

KLÍČOVÁ SLOVA

Plánování cesty, mobilní robot, optimalizace pomocí mravenčích kolonií.

KEYWORDS

Path planning, mobile robot, ant colony optimization.

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně dle pokynů vedoucího diplomové práce a s použitím uvedené literatury.

V Brně 27.5.2011

Bibliografická citace mé práce:

SEDLÁK, V. *Plánování cesty mobilního robotu pomocí mravenčích algoritmů.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2011. 64 s. Vedoucí diplomové práce RNDr. Jiří Dvořák, CSc..

PODĚKOVÁNÍ

Rád bych poděkoval RNDr. Jiřímu Dvořákovi za cenné podněty a rady k této práci.

Obsah:

1 ÚVOD

Tato práce se zabývá plánováním cesty a tvorbou trajektorie mobilního robotu. Cílem plánování je nalezení nekolizní cesty daným prostředím za stanovených omezujících podmínek. Ačkoli obecná formulace problému plánování cesty robota může být velmi snadná, řešení tohoto problému se musí vypořádat s mnoha komplikovanými omezeními. Zdroji omezení mohou být parametry robotu, prostředí ve kterém se má robot pohybovat, případně čas, během kterého musí být zvolený algoritmus pro plánování cesty schopen nalézt řešení. Plánování cesty robota se uplatňuje v širokém spektru oborů, od průmyslových aplikací, přes průzkum (jak civilní, tak vojenský), lékařství až po domácí aplikace v podobě úklidových robotů aj.

Mravenčí algoritmy jsou pravděpodobnostní algoritmy rojové inteligence inspirované chováním reálných mravenců při hledání potravy. Jedná se o kolonie jednoduchých umělých mravenců, kteří spolupracují na konstrukci řešení daného problému, přičemž využívají nepřímou komunikaci. Tato komunikace se uskutečňuje změnou pracovního prostředí agenta – ukládáním komunikační látky – feromonu. Nejpoužívanějšími algoritmy ze skupiny mravenčích algoritmů jsou algoritmy ze skupiny optimalizace pomocí mravenčích kolonií. Tato skupina algoritmů byla použita pro řešení širokého spektra optimalizačních problémů. Jednou z mnoha aplikací optimalizace pomocí mravenčích kolonií je i plánování cesty robota.

Cílem této práce je popsat základní principy plánování cesty robotu coby součást navigace a dále popsat principy algoritmů ze skupiny optimalizace pomocí mravenčích kolonií, navrhnout a implementovat metody plánování cesty pomocí zmíněných algoritmů a provést srovnávací a ověřovací experimenty.

Tato práce vychází z práce [15], kterou rozšiřuje o další algoritmy popsané v literatuře [2] – [19]. Plánování cesty je realizováno na grafové struktuře vytvořené nad statickým modelem prostředí s polygonálními překážkami. Zvolená reprezentace prostředí umožňuje s jistými omezeními modelovat i komplikované reálné prostředí. Navržené algoritmy musí být schopny nalézt optimální řešení, případně řešení blízko optimu. Kvalita řešení je ovlivňována parametry nastavení a doplňkovými procedurami a právě vliv těchto parametrů a procedur ve vztahu ke kvalitě řešení je zkoumán.

2 PLÁNOVÁNÍ CESTY ROBOTA

Jak uvádí práce [21] , plánování cesty je součástí navigace – definované jako určení pozice v prostoru a nalezení vhodné cesty pro dosažení cíle. Podle práce [20] je cílem navigace robota přesun z počáteční pozice do pozice koncové, přičemž obecný problém navigace lze popsat třemi otázkami:

- Jaká je startovní pozice? Odpovědí na tuto otázku je lokalizace robota v daném prostředí. Problematice určení pozice se velmi detailně věnují práce [20] a [22].
- Co je cílem? Pro splnění úkolu daného úkolu je nutné určit cílový stav, respektive rozpoznat dosažení cílového stavu.
- Jak dosáhnout cíle? Pokud jsou známy odpovědi na předchozí dvě otázky, redukuje se problém na nalezení takové posloupnosti akcí, které zajistí dosažení cílového stavu ze stavu počátečního při stanovených omezujících podmínkách.

Plánování cesty (tedy odpověď na otázku, jak dosáhnout cíle) může být lokální nebo globální, případně kombinované. Globální plánování hledá cestu daným prostředím před započetím pohybu robota, zatímco lokální plánování probíhá během pohybu robota. Při lokálním plánování musí robot rozpoznat překážky a v prostředí zvolit nekolizní trasu. Kombinovaný způsob hledá nejprve cestu daným prostředím globálně a po zahájení pohybu reaguje na případné změny prostředí a lokálně upravuje cestu [15].

Formálně lze problém plánování cesty popsat jako procházení stavů $x \in X$, kde X je množina všech stavů, označovaná jako stavový prostor. Cílem je nalezení takového stavu x, že $x \in X_G$, $X_G \subset X$, kde X_G je množina přípustných koncových stavů. Pro každý ze stavů $x \in X$ je určena množina přípustných přechodových akcí $U_{(x)} = \{u_1, \ldots, u_N\}$ taková, že libovolná přechodová akce $u \in U(x)$ použitá na aktuální stav x jej změní na stav x'. Množina $U = \bigcup_{x \in X} U_{(x)}$ představuje všechny možné akce. Jak dále uvádí práce [21] a [23], cílem je nalezení posloupnosti přechodových akcí, které transformují počáteční stav x_1 na koncový stav x_g z množiny X_G.

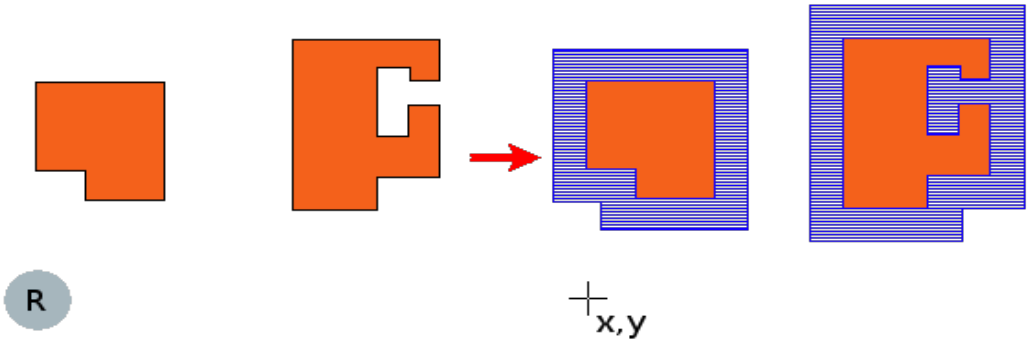
2.1 Pracovní a konfigurační prostor robota

Pro robot je určující prostor, kde plní dané úkoly. Práce [23] tento prostor označuje jako pracovní prostor *W,* který lze reprezentovat jako n-rozměrný euklidovský prostor $\mathbb{R}^n$. Pracovní prostor obsahuje objekty, které omezují pohyb robota – překážky. Ty lze dělit na statické, tedy takové, které nemění v čase svoji polohu či velikost, a dynamické, jejichž vlastnosti musí robot analyzovat až v době setkání s překážkou. V pracovním prostoru se může robot nacházet v určitým konfiguracích. Konfigurace je dána jako vektor, jehož složky jednoznačně určují polohu a orientaci robota v konfiguračním prostoru, dimenze tohoto vektoru je stejná jako počet stupňů volnosti robota. Všechny konfigurace robota definují konfigurační prostor *C.* V konfiguračním prostoru *C* lze vymezit oblast C_{free} – volný konfigurační prostor, kde jsou konfigurace robota přípustné. Podle práce [23] je C_{free} množinou všech konfigurací nekolidujících s překážkami a splňujících omezení kladená na robot. Další oblastí konfiguračního prostoru *C* je kolizní oblast C_{obs} – prostor všech konfigurací robota, při kterých dochází ke kolizi robota s některou překážkou nebo porušení omezení kladených na pohyb robotu[23]. Literatura dále uvádí možnost redukce konfiguračního prostoru, jak naznačuje obr. 1 , a to expanzí překážek v daném pracovním prostoru o velikost robota. Tato expanze umožňuje redukci robota do bodu a ve výsledku tedy dochází ke zjednodušení plánování cesty robota.

Konfigurační prostor s expandovanými překážkami umožňuje redukci plánování cesty robota na plánování pohybu bodu uvnitř volného konfiguračního prostoru. Pokud má plánování cesty v takto vzniklém konfiguračním prostoru proběhnout úspěšně, je nutné, aby startovní i cílové konfigurace patřily do volného konfiguračního prostoru.

2.2 Metody pro plánování cesty

Jak uvádí práce [15],[17],[23], existuje množství metod pro plánování cesty robota. Většina těchto metod má dvě hlavní části. Nejprve probíhá předzpracování, kdy je diskretizovaný pracovní prostor převeden na konfigurační prostor, a volný konfigurační prostor je poté popsán grafovou strukturou či funkcí. Plánování cesty se poté redukuje na hledání cesty grafem. Druhou částí plánování cesty, tzv. dotazovací částí, je poté užití některého z algoritmů pro hledání cesty grafem.



Obr. 1 Redukce robotu R a expanze překážek

2.2.1 Exaktní metody plánování cesty

Metoda plánování cesty je exaktní (také označovaná jako kompletní resp. úplná), pokud garantuje nalezení cesty mezi dvěma konfiguracemi za předpokladu existence této cesty. V opačném případě oznámí exaktní metoda, že cesta mezi danými konfiguracemi neexistuje.

2.2.2 Předzpracování pracovního prostoru

Vzhledem k velkému množství metod plánování cesty a jejich různých modifikací uvádím jen ty nejzákladnější metody předzpracování podle [23]:

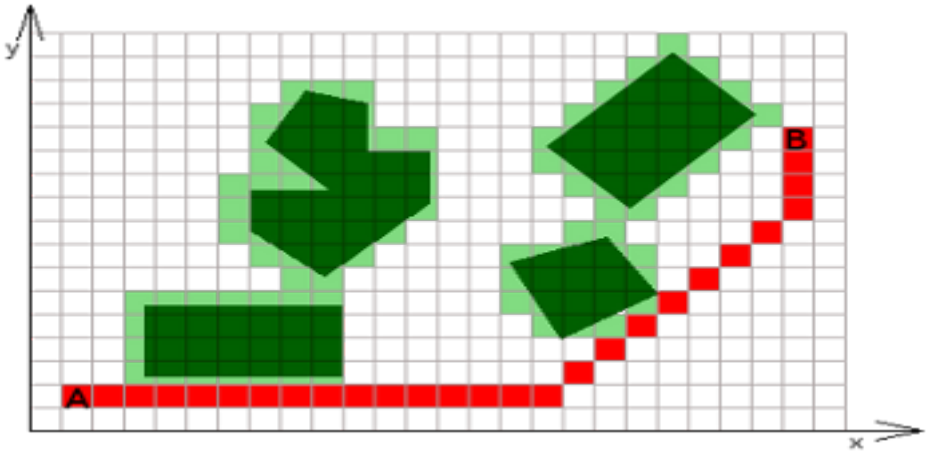
- Metody rozkladu do buněk
- Metody mapy cest
- Metody potenciálových polí

Rozklad do buněk

Princip těchto metod spočívá v rozkladu prostředí do buněk. Pro každou z buněk pracovního prostoru se určí, zda obsahuje volný prostor nebo překážku. Na základě těchto informací je vytvořen graf sousednosti. Metodu rozkladu do buněk lze použít jako exaktní nebo aproximativní.

Aproximativní rozklad do buněk

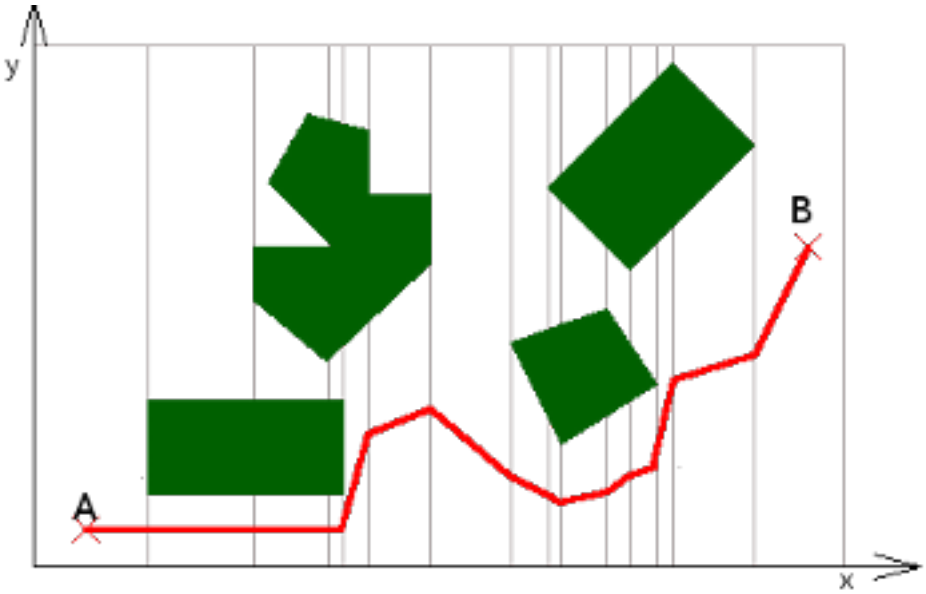
Metody aproximativního rozkladu do buněk rozkládají celý pracovní prostor do buněk stejného tvaru (nejčastěji čtvercového). V případě, že buňka obsahuje alespoň jeden bod překážky, je označena jako obsazená a není použita pro konstrukci grafové struktury. Velikost buněk může být jak konstantní, tak proměnná a od velikosti buněk se také odvíjí výpočetní náročnost této metody. Metody plánování cesty, které používají aproximativní rozklad do buněk, nejsou exaktní, protože se může stát, že nenajdou existující cestu. Příčinou může být vlastnost naznačená výše, kdy se za překážku považuje buňka, jejíž prostor překážka nevyplňuje zcela.



Obr. 2 Pracovní prostor rozložený aproximativní metodou do buněk o konstantní velikosti

Exaktní rozklad do buněk

Metody exaktního rozkladu rozkládají volný prostor do jednoduchých nepřekrývajících se buněk. Pro usnadnění výpočtu bývá volen tvar trojúhelníků, případně lichoběžníků. Jednoduchý tvar buněk je volen s ohledem na výpočetní náročnost. Grafová struktura je v případě exaktního rozkladu tvořena body přechodu mezi sousedními buňkami, startovními a cílovými body a spojnicemi mezi těmito body. Metody plánování cesty využívající exaktní rozklad do buněk jsou exaktní, naleznou tedy řešení, pokud existuje, v opačném případě ověří, že řešení neexistuje.



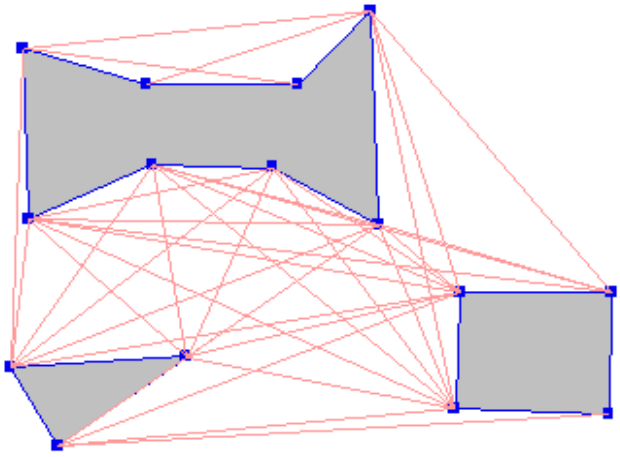
Obr. 3 Exaktní rozklad do lichoběžníkových buněk

Metody mapy cest

Metody využívající mapy cest vytváří grafovou strukturu (tzv. roadmapu), která reprezentuje volný pracovní prostor. Hrany grafu představují cesty, kudy se může robot pohybovat. K nalezení cesty v takto reprezentovaném prostředí se používá některý z algoritmů pro hledání cesty grafem. Metody mapy cest lze dělit na deterministické a pravděpodobnostní. Metody využívající mapy cest jsou exaktní, tedy naleznou řešení pokud existuje, v opačném případě ověří, že řešení neexistuje.

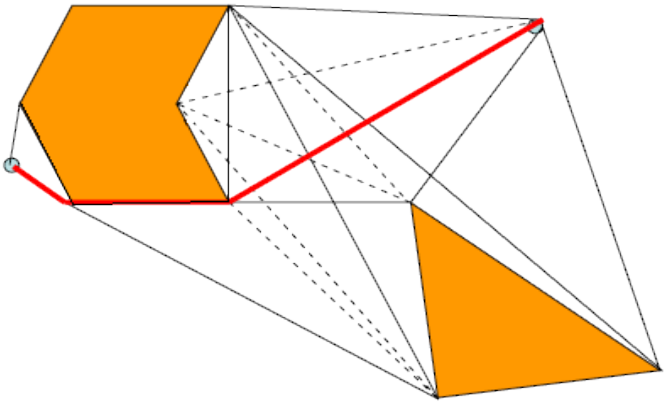
Graf viditelnosti

Vrcholy grafu viditelnosti představují startovní a cílové body a vrcholy překážek. Hrany grafu představují ty spojnice vrcholů, které neprotínají žádnou z překážek, tedy jsou vzájemně viditelné. Podmínkou pro použití grafu viditelnosti je polygonální reprezentace překážek. Pokud mají překážky nepolygonální tvar, je nutné je před vytvořením grafové struktury nahradit polygonem.



Obr. 4 Graf viditelnosti

Pro pracovní prostředí s velkým počtem vrcholů vyvstává problém s časovou náročností konstrukce grafu. Časovou náročnost lze snížit konstrukcí redukovaného grafu viditelnosti – tzv. grafu tečen. Jak název napovídá, hrany tohoto grafu jsou tečnami vrcholů. Tento typ grafu lze zkonstruovat i v prostředí s nepolygonálními překážkami. Snížení počtu hran grafu vede k rychlejšímu prohledávání grafu a tedy rychlejšímu nalezení cesty, pokud cesta existuje.

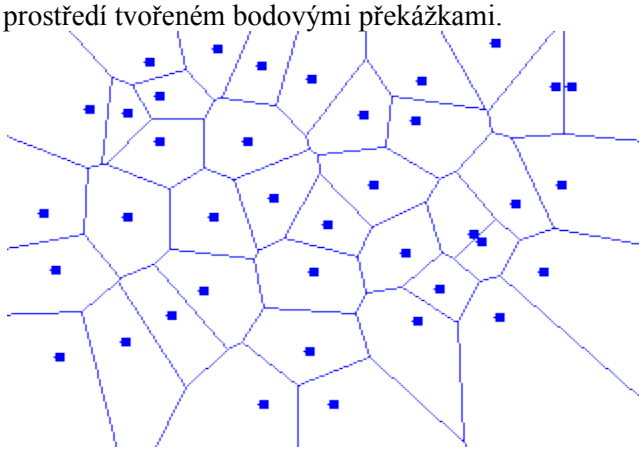


Obr. 5 Graf tečen, čerchovaně nepoužité hrany z grafu viditelnosti

Voronoiův diagram

Jak uvádí práce [15], Voronoiův diagram je geometrická struktura v rovině tvořená body o stejné vzdálenosti od jedné nebo více překážek. Body společně tvoří hrany diagramu, po kterých se robot může pohybovat bez rizika kolize s překážkou. Z diagramu je vytvořen graf, jehož vrcholy jsou body o stejné vzdálenosti od třech nebo více překážek a hrany jsou úsečky, tvořené body o stejné vzdálenosti od dvou překážek. Do grafu je třeba zahrnout první a poslední úsek cesty, které vzniknou napojením startu a cíle na nejbližší hrany Voronoiova diagramu.

Řešením je nalezení nejkratší cesty grafem. Na obr. 6 je ukázka Voronoiova diagramu ve zjednodušeném modelu prostředí tvořeném bodovými překážkami.



Obr. 6 Voronoiův diagram v prostředí s bodovými překážkami

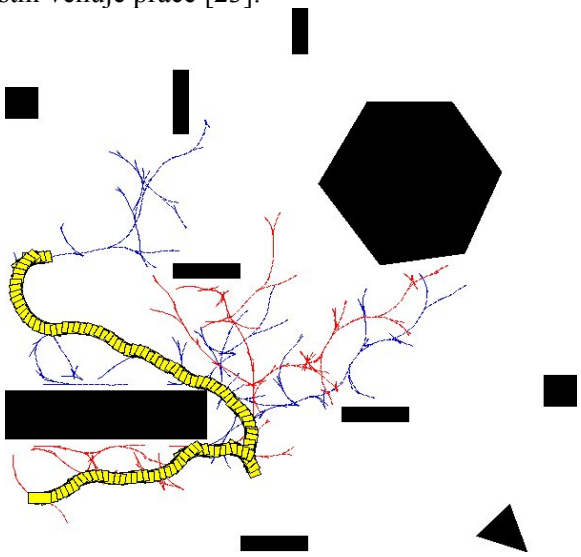
Pravděpodobnostní metody mapy cest

Algoritmy z této skupiny jsou dvoufázové. První je tzv. učící fáze, kdy probíhá tvorba grafové struktury. Vrcholy grafu jsou náhodně vybrané nekolizní konfigurace robota a hranami jsou nekolizní spojnice náhodně vygenerovaných vrcholů. První fáze běhu algoritmu pokračuje tak dlouho, dokud není dostatečně popsán volný konfigurační prostor C_{free}.

Druhá fáze spočívá v připojení počáteční a cílové konfigurace ke grafové struktuře a následném hledání cesty. Pokud algoritmus pravděpodobnostní mapy cest nenalezne řešení, nemusí to znamenat, že cesta neexistuje, protože pokrytí v první fázi nebylo dostatečné. Podle práce [23] může být dalším postupem v takové situaci opakování první fáze. Jak dále zmíněná práce uvádí, pravděpodobnostní mapy cest fungují dobře v prostorech se statickými překážkami, kdy stačí provést jedenkrát učící fázi a dle potřeby opakovat dotazovací fázi, tedy provést hledání cesty grafem.

Pravděpodobnostní stromy

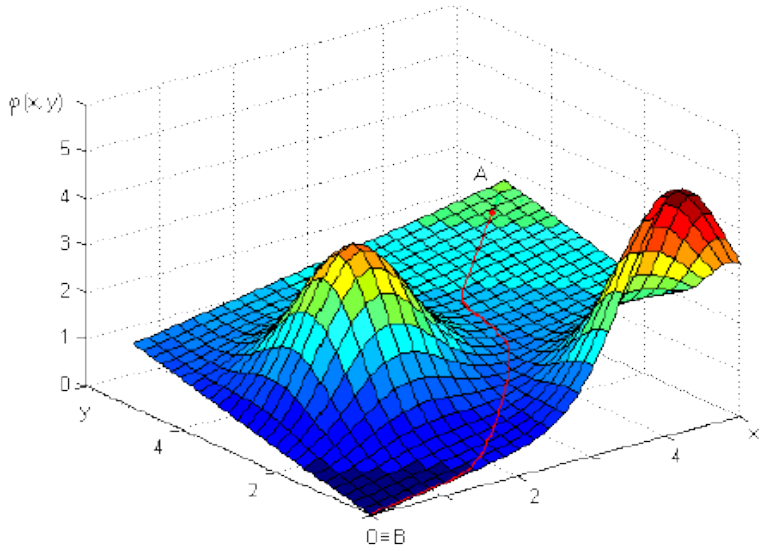
Hlavní myšlenkou těchto metod je rychlé inkrementální prohledávání konfiguračního prostoru, vytváření tzv. prohledávacího stromu. Vytváření stromu probíhá tak, že každé iteraci se stávající strom rozroste o náhodně vygenerovanou konfiguraci. Vrcholy stromu tedy představují konfigurace robota ve volném konfiguračním prostoru C_{free} a hrany představují akce, které vedou k dosažení daných konfigurací. Více se problematice pravděpodobnostních stromů a plánování cesty pomocí rychle-rostoucích pravděpodobnostní věnuje práce [23].



Obr. 7 Pravděpodobnostní stromy použité pro plánování cesty robota[24]

Potenciálová pole

Jak uvádí práce [23], hlavní myšlenka použití potenciálových polí je v pokrytí celého pracovního prostoru robota potenciálovým polem, které přiřadí každému bodu prostoru potenciál φ(x,y). Startovní bod robota má vyšší potenciál než cílový bod, překážky mají potenciál vyšší než jejich okolí. Robot se v takto reprezentovaném prostředí pohybuje ve směru největšího obráceného gradientu potenciálové funkce.



Obr. 8 Potenciálové pole [23]

2.2.3 Algoritmy pro plánování cesty

Po předzpracování, tj. vytvoření vhodné grafové struktury dostatečně popisující volný konfigurační prostor, se na vytvořenou grafovou strukturu aplikuje některý z algoritmů pro hledání cesty grafem. Existuje množství algoritmů, volba konkrétního algoritmu je otázkou konkrétní aplikace a požadavků na přesnost či rychlost hledání atd. Přehled různých algoritmů pro plánování cesty robota uvádí práce [21],[17].

2.3 Holonomní a neholonomní robot

Podle práce [23] lze roboty dělit podle omezení pohybu na holonomní a neholonomní. V klasické mechanice může být systém definován jako holonomní, pokud všechna jeho omezení jsou holonomní. Holonomní omezení jsou taková, která lze vyjádřit jako funkce $f(x1, x2, x3, ..., xn , t) = 0$, tj. omezení závisí pouze na souřadnicích systému a času. Omezení nezávisí na rychlosti nebo hybnosti systému.

Holonomnost v robotice je vyjadřována jako poměr řiditelných stupňů volnosti vůči jejich celkovému počtu. Pokud je počet řiditelných stupňů volnosti stejný jako jejich počet, pak je robot holonomní. Naopak pokud je počet řiditelných stupňů volnosti menší než celkový počet stupňů volnosti, pak je robot neholonomní. Podle práce [21] je robot neholonomní tehdy, jsou li na něj kladena diferenciální omezení, která nejsou plně integrovatelná (nemohou být převedena do tvaru nezahrnující derivaci).

Příkladem neholonomního robotu je auto, jehož pohyb do stran je omezen natočením předních kol. Plánování cesty neholonomního robota se potýká s jistými specifiky, kterým se více věnují práce [21] a [23].

2.4 Plánování cesty více robotů

Případy, kdy je nutné plánovat cestu více robotů současně, jsou stále častější. Současné použití více robotů může být výhodné v případě, kdy zadaný úkol přesahuje možnosti jednoho robota nebo možnost použití více robotů současně vede ke zkrácení doby řešení problému. Příkladem aplikace více robotů současně může být situace, kdy dva nebo více robotů přesouvá společně nějaký náklad pracovním prostorem, jiným příkladem aplikace více robotů může být použití několika bezpilotních letounů v určité oblasti, kdy cílem je dosáhnout maximálního pokrytí dané oblasti a současně minimální spotřeby paliva.

Použití více robotů současně má také jistá úskalí. Především se jedná o nebezpečí kolizí, jak uvádí práce [3], předcházení kolizí v systému s více roboty je základním problémem.

Jak dále uvádí práce [3], metody pro plánování cesty více robotů je možné rozdělit do dvou skupin podle přístupu k řešení problému:

- Centralizovaný přístup, kdy existuje jen jeden konfigurační prostor pro všechny roboty a plánování cesty se uskutečňuje současně.
- Decentralizovaný přístup, kdy je plánována cesta pro každého robota zvlášť a dodatečně jsou řešeny kolize.

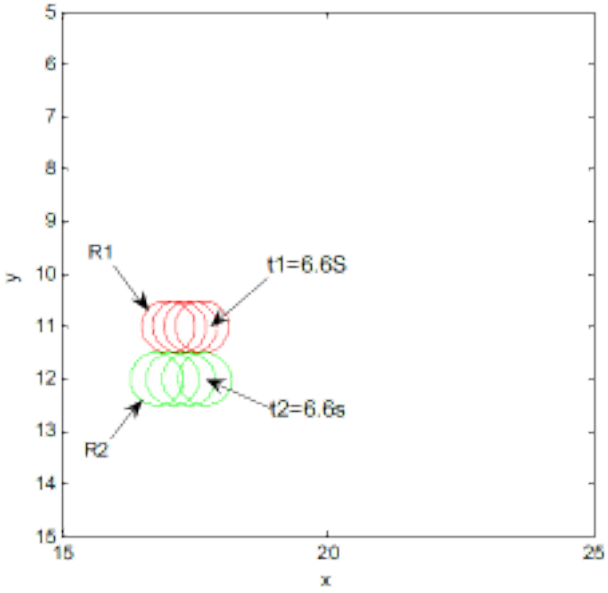
Oba zmíněné přístupy mají několik společných prvků. Jedná se především o nutnost řešit kolize, systém předcházení kolizí a potřeba úpravy trajektorie, event. přeplánování cesty. Dalším z problémů spojených s plánováním cesty více robotů, je rozvrhování úkolů pro jednotlivé roboty.

Výše uvedená práce uvádí přístup pro plánování cesty vycházející z druhého, decentralizovaného přístupu. Tento přístup předpokládá, že pro každého z n robotů je nalezena cesta daným prostorem. Tyto cesty jsou poté převedeny na trajektorie, tedy výstup z plánování cesty je doplněn o další parametr - čas , v závislosti na parametrech robota. Porovnáním trajektorií jsou nalezeny případné kolize a tyto kolize jsou řešeny. Práce [3] odkazuje na různé strategie řešení konfliktů, jako např. použití prioritních pravidel, náhodné zastavování robotů, aj. Řešení použité konfliktu na pozici P naznačené v [3]vychází ze dvou prioritních pravidel:

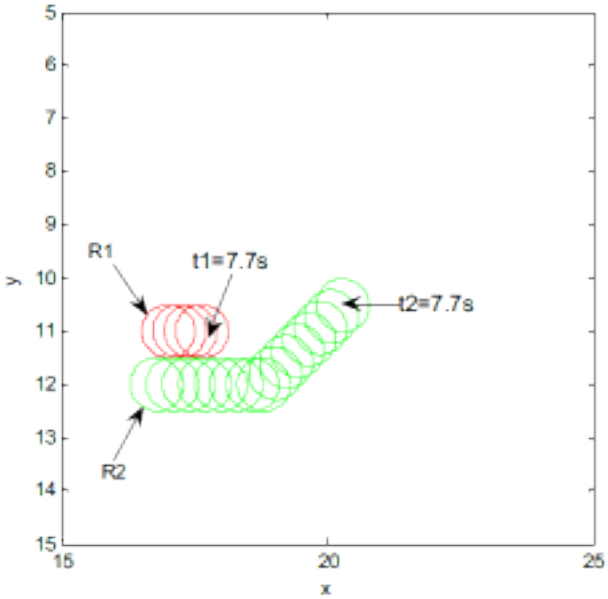
1. Pokud se robot R_i pohybuje směrem k pozici P a detekuje, že daná pozice je obsazena robotem R_j, zastaví se (event. zpomalí) a na pozici P se přesune až poté co ji robot R_i opustí.
2. Pokud se roboti R_i a R_j současně přibližují k volné pozici P, je uplatněno předem dané prioritní pravidlo, tedy robot s nižší prioritou zpomalí či zastaví a v pohybu na pozici

P pokračuje až v okamžiku, kdy robot s vyšší prioritou pozici P opustí.

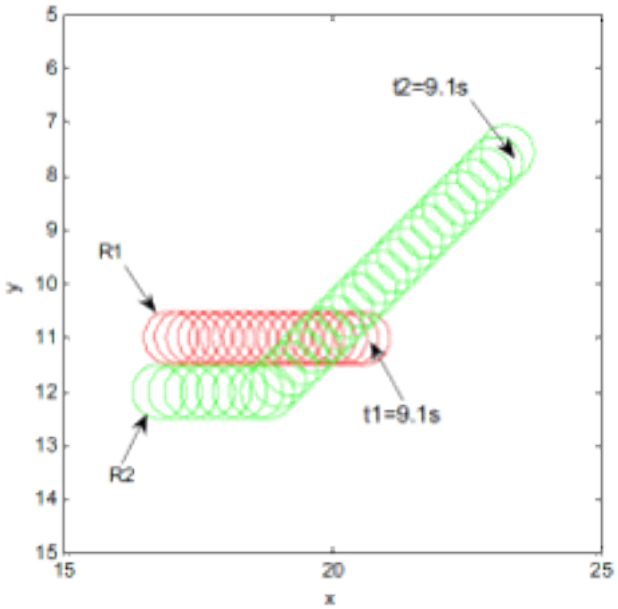
Použití těchto pravidel je naznačeno na obr. 9 – obr. 11 . Tento přístup k plánování cesty zaručí sice nekolizní prostup daným prostředím pro všechny roboty, ale nezaručí minimalizaci času potřebného pro splnění všech úkolů. V případě, že cílem plánování cesty je nejen zajištění nekolizního pohybu v pracovním prostředí robota, ale také minimalizace časových nároků, lze podle práce [25] problém plánování cesty robota chápat jako rozvrhovací problém. Jak práce [25] dále uvádí, exaktní řešení takového problému může být časově velmi náročné, zmíněná práce uvádí pro řešení daného problému přístup založený na kombinaci prioritních pravidel a náhodného výběru strategie pohybu robotů z množiny možných řešení. Podle uvedené práce dosahuje algoritmus založený na zmíněném přístupu velmi dobré kvality řešení daného problému.



Obr. 9 Plánování cesty 2 robotů, použití prioritních pravidel[3]



Obr. 10 Plánování cesty 2 robotů, použití prioritních pravidel[3]



Obr. 11 Plánování cesty 2 robotů, použití prioritních pravidel[3]

3 MRAVENČÍ ALGORITMY

Mravenčí algoritmy představují skupinu evolučních algoritmů patřících mezi algoritmy rojové inteligence, které jsou inspirované chováním sociálního hmyzu – v tomto případě mravenců. Jedná se o pravděpodobnostní algoritmy, které využívají samoorganizační schopnosti kolonie umělých agentů, přičemž inspirací pro vznik těchto algoritmů byla schopnost vysoce koordinované činnosti mravenčích kolonií.

Jednou z nejznámějších a nejpoužívanějších skupin těchto algoritmů je optimalizace pomocí mravenčích kolonií (také uváděna jako ant colony optimization, ACO), jejíž inspirací bylo chování mravenců při hledání potravy. Jak uvádí práce [10] a [12], první algoritmus z rodiny optimalizačních mravenčích algoritmů(viz § 3.5) vznikal na počátku 90. let 20. stol. v rámci doktorské práce M. Doriga . Původně byly mravenčí algoritmy použity pro hledání cesty grafem, postupně ovšem byly tyto algoritmy použity pro řešení mnoha různých problémů, jak je uvedeno v kapitole 3.6 .

3.1 Reálné mravenčí kolonie

Inspirací pro vznik mravenčích algoritmů bylo chování reálných mravenčích kolonií. Některé druhy mravenců mají schopnost najít nejkratší cestu mezi hnízdem a zdrojem potravy, a to i přesto, že mravenci z těchto kolonií jsou téměř slepí. Dalším zajímavou vlastností mravenčích kolonií, a kolonií sociálního hmyzu obecně, je fakt, že schopnosti a možnosti jednotlivých členů kolonií jsou velmi omezené, ale schopnosti kolonie jako celku jsou vysoce koordinované a komplexní. Dalším charakteristickým rysem mravenčích kolonií je, že mravenci mezi sebou nekomunikují přímo, ale změnou prostředí ve kterém se pohybují. Tato změna je realizována ukládáním feromonových stop, na které reagují další členové kolonie, a právě tento způsob komunikace je podle práce [13] klíčem k samoorganizačním schopnostem mravenčích kolonií. V mravenčích koloniích se taktéž uplatňuje princip pozitivní zpětné vazby. Výše zmíněný způsob komunikace se v literatuře označuje jako tzv. stigmergie (ev. stigmergy). Feromonové stopy se po uložení postupně odpařují, což způsobuje částečné zapomínání, ale také zabraňuje konvergenci řešení.

3.1.1 Pohyb a chování mravenců

Jak uvádí literatura [1]–[12], pohyb reálných mravenců je pravděpodobnostní – pravděpodobnost, že mravenec se rozhodne pro jistou cestu, je dána množstvím feromonu uloženým jinými mravenci na této cestě. Tento pravděpodobnostní pohyb je také jedním ze základních kamenů samoorganizace mravenčích kolonií. Pro vysvětlení principů fungujících v reálných mravenčích koloniích poslouží obr. 12 .

Na obr. 12 je zobrazen zdroj potravy (P) a hnízdo (H), nalezení cesty mezi těmito dvěma pozicemi lze popsat 4 kroky [15].

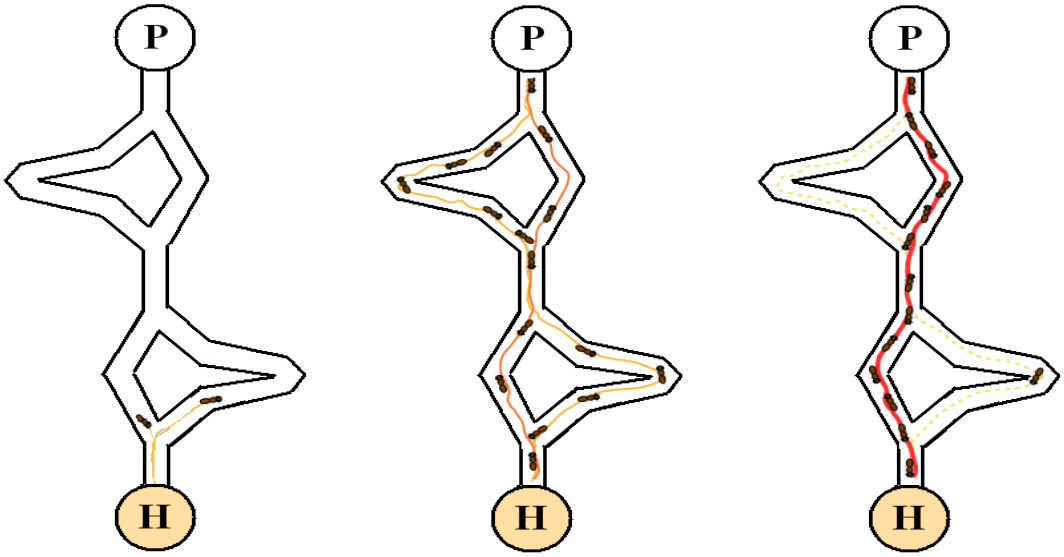
a) Mravenci hledají cestu ke zdroji potravy (P) a náhodně prohledávají okolí mraveniště (H). V okamžiku, kdy mravenec najde zdroj potravy (P), sebere potravu a začne hledat cestu zpět k mraveništi. Cestou zpět od zdroje potravy k hnízdu ukládá mravenec na cestu určité množství feromonu, na který reagují ostatní mravenci.
b) Pokud mravenec při hledání cesty (jak ke zdroji potravy, tak zpět do hnízda) dorazí do bodu, kde se cesta rozděluje do více větví, rozhodne se pro *i*-tou cestu s pravděpodobností $P_i(t)$

$$P_{(i)}(t)=\frac{(S_{(i)}(t)+k)^{h}}{\sum_{j=1}^{n}(S_{(j)}(t)+k)^{h}} \qquad (1)$$

kde $S_i(t)$ je intenzita feromonové stopy na větvi i, n je počet možných pokračování cesty, h a k jsou konstanty. Dle [10] odpovídají experimentům s reálnými mravenčími koloniemi hodnoty $k = 20$ a $h = 2$.

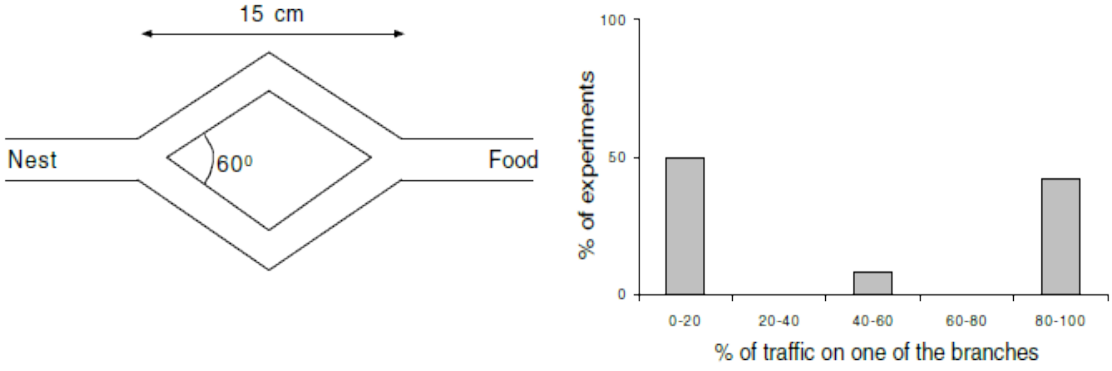
c) Na počátku není nikde mezi zdrojem potravy a hnízdem uložen feromon a všechny existující cesty se mravencům jeví stejné, proto mravenci hledají volí cestu náhodně.
d) Z obr. 12 je patrné, že existuje několik spojnic hnízda a zdroje potravy. Pokud na

počátku (tedy v okamžiku, kdy nikde na cestě ještě nejsou feromonové stopy) některý z mravenců (pro zjednodušení ho označme jako *M*) zvolí delší variantu cesty k potravě, dorazí k potravě později než jedinci, kteří zvolili kratší cestu. Jak je uvedeno v a), mravenci cestou nazpět do hnízda ukládají feromonovou stopu, proto jedinec M cestou nazpět do hnízda nebude cestu hledat náhodně, ale bude s pravděpodobností podle rovnice (1) preferovat cestu, kterou před ním prošli mravenci, kteří cestou k potravě použili kratší cestu. Mravenec *M* (a každý další mravenec navracející se do hnízda) uloží jisté množství feromonu na cestu, čímž se tato cesta stane atraktivnější pro ostatní členy kolonie. Čím více mravenců projde jednou cestou, tím atraktivnější se tato cesta bude jevit ostatním členům kolonie a tuto cestu bude postupně využívat více mravenců až nakonec zcela převládne. Právě tento princip, kdy malá počáteční odchylka vyústí ve velkou změnu a ustálení systému v novém stavu, je již zmíněnou pozitivní zpětnou vazbou.



Obr. 12 Nalezení nejkratší spojnice mezi hnízdem a zdrojem potravy[15]

Literatura [6], [10] a [11] uvádí experimenty provedené s reálnými mravenčími koloniemi s dvojitým mostem, a to s případy stejné a rozdílné délky větví. Pokud mezi hnízdem a zdrojem potravy existují dvě stejně dlouhé cesty (viz obr. 13), dojde po určitém čase k ustálení pohybu pouze po jedné z nich. Důvod takového chování je naznačen výše (viz a) – d)), na počátku není na žádné z možných cest feromonová stopa a mravencům se tedy všechny cesty jeví stejné a proto volí cesty náhodně. Pokud se na některou z cest vydá více mravenců, dojde k uložení většího množství feromonu a tato cesta bude mravenci upřednostňována. Pozitivní zpětná vazba pak zajistí, že všichni mravenci začnou používat jen jednu z cest [15].



Obr. 13 Experiment s dvojitým mostem [10]

Případ, kdy mezi hnízdem a zdrojem potravy existují minimálně dvě spojnice o rozdílných délkách, popisují body a) – d).

Reálné mravenčí kolonie jsou schopny reagovat na změnu prostředí, příkladem je umístění překážky na existující cestu. Pokud mravenec sleduje feromonovou stopu a dorazí až k překážce dělící tuto stopu, nedojde k ničemu jinému, než k postupu podle bodů a) – d), tedy postupnému prohledávání okolí, nalezení nové cesty a postupnému ustálení řešení.

Dalším zajímavým prvkem v chování mravenčích kolonií je podle [10] situace, kdy je po ustálení pohybu mravenců po určité cestě mezi hnízdem a zdrojem potravy této kolonii „nabídnuta" alternativa – kratší spojnice hnízda a zdroje potravy. Podle literatury [10] paradoxně nedojde k ustálení pohybu na této kratší cestě, ale kolonie zůstane u původního řešení a jen sporadicky se vyskytnou mravenci, kteří zvolí novou cestu. Literatura nabízí dvě vysvětlení, proč k této situaci dochází:

1. I když dojde k nalezení nové cesty, vysoká hodnota feromonu na původní cestě v kombinaci s pozitivní zpětnou vazbou zabrání tomu, aby mravenci ve větší míře používali novou cestu.
2. Odpařování feromonových stop probíhá velmi pomalu a v kombinaci s výše zmíněným nedojde k tomu, aby novou cestu zvolil dostatečný počet mravenců.

Jak dále literatura uvádí, pouze takové mravenčí kolonie mají schopnost nalézt nejkratší cestu ke zdroji potravy, kde mravenci ukládají feromonové stopy pouze cestou nazpět do hnízda. Důvod, proč mravenčí kolonie, kde mravenci ukládají feromonovou stopu i při hledání potravy, nemají schopnost najít nejkratší spojnici mezi hnízdem a potravou, je opět naznačen v bodech popisujících pohyb mravenců (viz a) – d) výše). Pokud se v prostředí vyskytne feromonová stopa ještě před nalezením zdroje potravy, může takováto stopa díky samoorganizační schopnosti kolonie a pozitivní zpětné vazby zcela převážit a to bez ohledu na její optimalitu.

3.2 Umělé mravenčí kolonie

Výše zmíněné vlastnosti reálných mravenčích kolonií se staly inspirací pro vznik mravenčích algoritmů, resp. optimalizaci pomocí mravenčích kolonií. Stejně jako reálné mravenčí kolonie, i kolonie umělých mravenců mají optimalizační schopnosti. Tyto dva typy kolonií jednoduchých agentů jsou si v některých ohledech velmi podobné. Jedná se především o kooperaci členů těchto kolonií na dosažení cíle a také fakt, že schopnosti obou typů kolonií jako celků vysoce převyšují schopnosti jednotlivých členů těchto kolonií.

Jak uvádí práce [13],[10], kolonie umělých mravenců se od reálných předloh liší především v těchto bodech:

- Umělí mravenci se pohybují v diskrétním přes konečnou množinu bodů.
- Umělí mravenci mají paměť, která zaznamenává provedené akce.
- Množství komunikační látky ukládané umělými mravenci může být funkcí kvality řešení.
- Odpařování komunikační látky zajišťuje ochranu před uvíznutím v lokálním extrému, odpařování také umožňuje zapomenout horší řešení a směřovat prohledávání do jiných míst.
- Umělí mravenci ukládají feromonovou stopu až po nalezení řešení.

Jak uvádí [6] a [10] , cílem mravenčích algoritmů není detailní simulace chování reálné mravenčí kolonie při hledání potravy, ale využití optimalizačních a samoorganizačních vlastností těchto kolonií pro řešení složitých optimalizačních problémů

3.3 Optimalizace pomocí mravenčích kolonií

Optimalizace pomocí mravenčích kolonií (také uváděna jako ant colony optimization – ACO) je optimalizační metoda inspirovaná schopnostmi reálných mravenčích kolonií. Touto optimalizační metodou lze, dle prací [10] a [12], řešit jakýkoli optimalizační problém, který lze převést na prohledávání grafu a pro který existuje nějaká konstruktivní heuristika.

Mravenci v ACO slouží jako stochastické konstrukční procedury, které vytváří řešení postupným přidáváním komponent do částečného řešení. Výběr přidávaných komponent ovlivňuje heuristická informace o řešeném problému, která umožňuje směřování výpočtu ke slibným řešením. Dále je výběr komponent ovlivňován zkušenostmi, které představují feromonové stopy a které byly

získány od počátku řešení.

Jak je uvedeno výše, algoritmy ze skupiny optimalizace pomocí mravenčích kolonií mohou řešit široké spektrum problémů, ovšem před aplikací ACO metaheuristiky na daný problém je nutné převést daný problém do takového tvaru, který umožní umělým mravencům konstruovat řešení daného problému.

Literatura [10] uvádí jako příklad problému řešeného pomocí ACO minimalizační problém daný jako trojice (S, f, Ω), kde S představuje množinu možných řešení, f je účelová funkce, pro kterou hledáme optimum. Funkce f přiřazuje každému řešení $s \in S$ hodnotu $f_{(s,t)}$. $\Omega_{(t)}$ je množina omezujících podmínek. Parametr t znamená, že účelová funkce a množina omezujících podmínek může být časově závislá. Cílem je nalezení globálního optima s^*.

Řešený kombinatorický optimalizační problém (S, f, Ω) je dále specifikován následujícími položkami.

- Konečná množina $C = \{c_1, c_2, \ldots, c_{Nc}\}$ komponent, kde Nc je počet komponent.
- Stavy problému $x = \langle c_i, c_j, \ldots, c_h, \ldots \rangle$, náleží do množiny všech stavů problému X. Každý ze stavů $x_i \in X$ je konečnou posloupností komponent c.
- Množina možných řešení S , $S \subseteq X$.
- Množina přípustných stavů $X' \subseteq X$, definovaná jako stavy, do kterých není nemožné se dostat za omezujících podmínek Ω. Jak uvádí [10], takováto definice ovšem nezaručuje nalezení řešení s takového, že $s \in X'$.
- Neprázdná množina S^* optimálních řešení, kde $S^* \subseteq X'$, $S^* \subseteq S$.
- Náklady $g(s,t)$ spojené s každým možným řešením $s \in S$. Většinou platí $g(s,t) \equiv f(s,t), \forall s \in S'$,kde $S' \subseteq S$ je množina vyhovujících možných řešení za omezení $\Omega(t)$.

Podle této formulace tvorba řešení probíhá tak, že v úplném grafu $Gc = (C,L)$, kde L je množina hran spojující komponenty C, provádí kolonie umělých mravenců náhodné přesuny po hranách grafu za omezujících podmínek $\Omega(t)$. Komponenty $c_i \in C$ a hrany $l_{ij} \in L$ mohou být spojeny s feromonovou stoupou $\tau_{(i,j)}$ a heuristickou informací $\eta_{(i,j)}$. Feromonová stopa představuje dlouhodobou paměť celého prohledávání a je vytvářena a aktualizována samotnými agenty. Heuristická informace na rozdíl od feromonové stopy představuje informaci o instanci problému. Právě feromonová stopa a heuristická informace jsou jediné dvě informace používané mravenci pro uskutečnění pravděpodobnostních pohybů v grafu.

Každý z agentů v kolonii umělých mravenců následující vlastnosti:

- Využívá graf $G_c = (C,L)$ pro hledání optimálního řešení $s^* \in S^*$.
- Využívá paměť M, která může být použita pro uložení informací o použité cestě, nebo tvorbu řešení, nebo zpětnou rekonstrukci cesty.
- Má počáteční stav x_s, a alespoň jednu ukončující podmínku e.
- Pokud ve stavu $x_r = \langle x_{r-1}, i \rangle$ není splněna žádná z ukončujících podmínek, tak se přesune do stavu $x_j = \langle x_r, j \rangle$ v okolí $N(x_r)$ stavu x_r. Pokud je splněna ukončující podmínka, agent se zastaví.
- Stav, kam se agent přesune je vybírán stochastickým rozhodovacím pravidlem. Pravděpodobnostní rozhodovací pravidlo je závislé na stavu feromonové stopy a heuristické informace, nebo paměti agenta, nebo omezujících podmínek.
- Při přidání komponenty c_j do aktuálního stavu může aktualizovat hodnotu feromonové stopy τ příslušné k dané komponentě nebo spojení s touto komponentou.
- Po dokončení řešení se agent dokáže přesunout stejnou cestou nazpět a dokáže aktualizovat feromonové stopy na použitých komponentách.

Jak dále uvádí [10], v umělé mravenčí kolonii hledají řešení daného problému současně všichni agenti nezávisle na sobě. Schopnosti jednotlivých agentů jsou přitom dostatečné pro nalezení řešení (byť nedobrého). Podle výše uvedené práce je ale pro nalezení dostatečně kvalitního řešení nutná spolupráce a interakce celé kolonie umělých mravenců. Přitom tato spolupráce a interakce je možná díky nepřímé komunikaci – tedy díky feromonovým stopám, které využívají a aktualizují členové kolonie. Podle [10] se jedná o distribuovaný proces učení, kde se nepřizpůsobují agenti řešící daný problém, ale agenti mění způsob, jakým je problém zprostředkováván zbytku kolonie a jak ho ostatní členové kolonie vnímají.

3.4 Metaheuristika ACO

Podle literatury [10],[12],[14],[16] lze kostru většiny ACO algoritmů popsat takto:

```
Procedure ACO _ MetaHeuristic
    initialize()
  do−while(not termination)
    generateSolutions()
    updatePheromone()
    deamonActions()
    end−while
end− procedure
```

Obr. 14 Obecná kostra ACO algoritmů

Před spuštěním hlavního cyklu probíhá inicializační procedura, kdy jsou mravenčí kolonii nastaveny parametry hledání a především dochází k inicializaci grafové struktury a nastavení počáteční hodnoty umělé feromonové stopy na hranách grafu.

Po inicializaci datových struktur potřebných k řešení daného problému jsou iterativně volány tři procedury:

- V první proceduře hlavního cyklu konstruuje *m* mravenců současně a asynchronně řešení, přičemž prochází sousední vrcholy grafové struktury zmíněné výše. Pohyb agentů je ovlivňován lokální heuristickou informací a intenzitou umělých feromonových stop na komponentách grafové struktury. Řešení konstruované v této etapě algoritmu ovlivňuje množství umělého feromonu ukládaného v další etapě.
- Ve druhé etapě hlavního cyklu probíhá aktualizace hodnot umělého feromonu na komponentách grafové struktury. Může se jednat o zvýšení hodnot, v případě ukládání feromonových stop, stejně tak i o snížení, v případě odpařování umělé feromonové stopy, popřípadě o kombinaci obou. Jak již bylo zmíněno, zvýšení hodnoty feromonu zvyšuje pravděpodobnost využití dané komponenty grafu členy mravenčí kolonie. Naopak odpařování feromonu je jistou formou zapomínání a dle [10] se jedná o možnost, jak omezit konvergenci řešení do lokálního optima, nebo jak přimět algoritmus prohledávat ještě neprozkoumané části grafové struktury.
- Procedura *deamonActions* je mnohdy rozšířením algoritmu. Většinou tato procedura provádí centralizovanou činnost, kterou není možné provádět na úrovni jednotlivých agentů. Jedná se například o doplnění prohledávání stavového prostoru, zvýhodnění některých nalezených řešení aj. .

Jak je patrné z obr. 14 , běh algoritmu je ukončen po dosažení stanovených podmínek ukončení. Této problematice se více věnuje § 3.8.1 .

3.5 Základní ACO algoritmy

Literatura uvádí množství variant ACO algoritmů, přičemž tyto algoritmy většinou rozšiřují některý z algoritmů uvedených v následujících podkapitolách. Rozšíření některého z ACO algoritmů je obecně možné ve více směrech. Lze upravit schéma ukládání feromonových stop, je možné předefinovat heuristickou informaci, případně lze upravit pravidla pro pohyb agentů po grafové struktuře. Dalším možným rozšířením ACO algoritmů je rozšíření procedury *deamonAction*s z § 3.4 , práce [14] navrhuje doplnění lokálního prohledávání a úpravu nejlepšího nalezeného řešení, práce [4] uvádí možnost komunikace a výměny informací mezi více koloniemi při řešení stejného problému aj. .

Dalším z možných přístupů k rozšíření ACO algoritmů je kombinace ACO algoritmu a některého z dalších optimalizačních algoritmů. Více se této problematice věnuje práce [6], kde je uvedena kombinace ACO a genetických algoritmů, dále pak kombinace ACO a simulovaného žíhání, jak uvádí [2].

Dále jsou uvedeny základní algoritmy ze skupiny ACO algoritmů ve formě, jakou mají pro řešení problému obchodního cestujícího nebo plánování trasy robota. Adaptace těchto algoritmů pro jiné problémy jsou uvedeny v literatuře [10] a [19].

3.5.1 Ant system

Většina dnes používaných algoritmů z rodiny ACO algoritmů navazuje nebo rozšiřuje původní z těchto algoritmů – ant system, který byl publikován na počátku 90. let jako součást doktorské práce M. Doriga. První řešené problémy tímto algoritmem byly spojeny s hledáním optimální cesty grafem.

Jak je uvedeno výše, umělí mravenci konstruují řešení procházením grafové struktury, kdy je jejich pohyb ovlivňován heuristickou informací a intenzitou feromonových stop. Pohyb mravenců v umělých mravenčích koloniích je stejně jako u reálných předloh pravděpodobnostní. Pravděpodobnost, že mravenec *k* použije při konstrukci řešení použije hranu spojující komponenty *i j* je dána rovnicí:

$$p^{k}_{(i,j)} = \frac{|\tau_{(i,j)}|^{\alpha} \cdot |\eta_{(i,j)}|^{\beta}}{\sum_{z \notin Tabu^{k}} |\tau_{(i,z)}|^{\alpha} \cdot |\eta_{(i,z)}|^{\beta}} \tag{2}$$

kde $\tau_{(i,j)}$ představuje intenzitu feromonové stopy na hraně *(i,j),* $\eta_{(i,j)}$ představuje heuristickou informaci spojenou s hranou *(i,j).* Proměnná *z* představuje všechny bezprostřední sousedy vrcholu *i*. Pro většinu problému je $\eta_{(i,j)}$uvažováno jako viditelnost vrcholu *j* z vrcholu *i* a je dáno vztahem:

$$\eta_{(i,j)} = \frac{1}{d_{(i,j)}} \tag{3}$$

kde *d(i,j)* je vzdálenost vrcholů *i* a *j,* event. délka hrany *(i,j).* Exponenty *α,β* určují důležitost intenzity feromonové stopy, resp. heuristické informace při výpočtu pravděpodobnosti cesty z *i* do *j*. $Tabu^{k}$ je oblast zakázaných stavů, většinou se jedná o vrcholy či stavy již navštívené během konstrukce řešení. Průběžné ukládání aktuálních stavů do paměti agenta může zamezit opětovnému přechodu do již navštíveného stavu a případnému zacyklení běhu algoritmu.

Po nalezení řešení následuje aktualizace feromonových stop (viz § 3.4 a práce [10], [12]). Ant system aktualizuje feromonové stopy v okamžiku, kdy všichni mravenci nalezli řešení, nebo ukončili hledání řešení. Aktualizace feromonových stop se děje ve dvou krocích. Prvním je snížení hodnoty feromonových stop na všech hranách grafu. Toto „odpařování“ feromonových stop je dáno rovnicí:

$$\tau_{(i,j)} = \rho \cdot \tau_{(i,j)} \tag{4}$$

kde $\tau_{(i,j)}$ je množství feromonu a $0 \leq \rho \leq 1$ je koeficient odpařování. Po realizaci odpařování feromonu dochází k ukládání feromonových stop na těch hranách, které byly použity pro konstrukci řešení. Pro Ant system je charakteristické, že všichni agenti, kteří nalezli řešení, zvyšují hodnotu feromonových stop na hranách, které využili pro konstrukci řešení. Pokud některý z agentů řešení nenalezl, na aktualizaci hodnot feromonových stop se nepodílí.

Výsledná intenzita feromonových stop je pak dána vztahem:

$$\tau_{(i,j)} = \tau_{(i,j)} + \sum_{k=1}^{m} \Delta \tau^{k}_{(i,j)} \tag{5}$$

kde $\Delta\tau_{(i,j)}^{k}$ představuje přírůstek intenzity feromonové stopy na hraně *(i,j)* uložený mravencem *k.* Přitom platí:

$$\Delta \tau^{k}_{(i,j)} = \begin{cases} \frac{1}{C^{k}} & \forall (i,j) \in T^{k} \\ 0 & \forall (i,j) \notin T^{k} \end{cases} \tag{6}$$

kde *Q* je konstanta, C^{k} je délka řešení zkonstruovaného agentem *k,* T^{k} je řešení zkonstruované mravencem *k.* Jak uvádí literatura [10],[12] a jak vyplývá z rovnic (5) a (6), hrany náležící do řešení s menší celkovou délkou cesty a hrany kudy prošlo více mravenců, mají při aktualizaci hodnot feromonu větší přírůstky a zvyšuje se tedy pravděpodobnost, že budou v dalších krocích běhu algoritmu opětovně vybrány pro konstrukci řešení. Jak je zmíněno výše, všichni agenti, kteří nalezli řešení, ukládají zvyšují hodnoty feromonových stop. Tento přístup může vést k horším výsledkům pro rozsáhlejší domény nebo prodloužení doby potřebné pro nalezení optimálního řešení, event. řešení blízko optimu.

Níže uvedené algoritmy představují vylepšení algoritmu ant system, v případě Ant colony system pak rozšíření.

3.5.2 Elitist ant system

Tento algoritmus byl publikován nedlouho po původním Ant systemu. Hlavní myšlenkou tohoto vylepšení původního algoritmu je posílení feromonové stopy na nejlepším nalezeném řešení od od spuštění běhu algoritmu. Podle literatury [10] představuje toto posílení feromonové stopy na hranách nejlepšího řešení ukázku použití procedury *deamonActions* z ACO metauheristiky (viz kapitola 3.4).

Pro tento algoritmus zůstávají v platnosti výše uvedené vlastnosti a rovnice { viz (2),(3),(4), (5),(6)} a je doplněna další vlastnost, a to již zmiňované posílení feromonové stopy na nejlepším nalezeném řešení podle rovnice:

$$\tau_{(i,j)} = \tau_{(i,j)} + \Delta\tau^{bs} \qquad \forall (i,j) \in T^{bs} \tag{7}$$

kde $\Delta\tau^{bs}$ je přírůstek intenzity feromonové stopy na hraně *(i,j)* patřící do nejlepšího nalezeného řešení T^{bs} . $\Delta\tau^{bs}$ je dáno rovnicí (8), kde L^{bs} je délka nejlepšího řešení, *e* je koeficient určující důležitost řešení. Podle [10] umožňuje vhodně zvolený koeficient *e* nalezení lepších řešení v kratším čase. Nevhodně zvolený koeficient *e* může naopak vést ke stagnaci řešení, či uvíznutí v lokálním optimu.

$$\Delta\tau^{bs}_{(i,j)} = \frac{e}{L^{bs}} \tag{8}$$

3.5.3 Rank-based ant system

Tento algoritmus (také uváděný jako AS_{rank}) představuje další vylepšení algoritmu Ant system. Principem vylepšení je stejně jako v případě Elitist AS úprava ukládání feromonových stop. Zatímco v původní verzi algoritmu Ant system ukládají feromonové stopy všichni mravenci, kteří nalezli řešení, v tomto algoritmu pouze omezený počet agentů ukládá feromonové stopy. Z rovnic popisujících chování algoritmu ant system zůstávají v platnosti (2),(3) a (4).

Ukládání feromonových stop probíhá následujícím způsobem. Po dokončení iterace jsou mravenci, kteří nalezli řešení, seřazeni podle kvality řešení a pouze *w-1* nejlépe ohodnocených mravenců ukládá feromonovou stopu. Dále ještě dojde k uložení feromonové stopy na dosud nejlepší řešení – bez ohledu na to, zda bylo nalezeno v aktuální iteraci či nikoliv. Zvýšení hodnot feromonových stop je možné popsat takto:

$$\tau_{(i,j)} = \tau_{(i,j)} + \sum_{r=1}^{w-1} (w-r)\,\Delta\tau^{r}_{(i,j)} + w\,\Delta\tau^{bs}_{(i,j)} \tag{9}$$

přičemž platí:

$$\Delta\tau^{bs}_{(i,j)} = \frac{1}{L^{bs}} \tag{10}$$

$$\Delta\tau^{r}_{(i,j)} = \frac{1}{L^{r}} \tag{11}$$

kde L^r je délka řešení *r*-tého Mravence, L^{bs} je stejně jako v § 3.5.2 délka nejkratšího doposud nalezeného řešení. Jak je patrné z rovnice (9), v tomto algoritmu je množství feromonu ukládaného mravenci násobeno váhou ležící v rozmezí $\langle w\ ,\ 1\rangle$, přičemž nejvíce feromonu je uloženo na doposud nejlepší nalezené řešení.

Jak uvádí [10], Rank-based ant system dosahuje při řešení většiny problémů obdobných výsledků jako elitist ant systém a lepších výsledků než původní verze algoritmu Ant system.

3.5.4 Max–min ant system

Jak je uvedeno v literatuře [10],[12],[13],[14], Max–min ant system (také uváděný pod zkratkou MMAS) se liší od původního algoritmu ve 4 hlavních bodech:

1. Využívá jen nejlepší řešení, resp. k ukládání feromonové stopy dochází pouze na hranách použitých pro konstrukci nejlepšího řešení. Odpařování feromonových stop je aplikováno stejně jako v původním Ant systemu.
2. Protože ukládání uvedené v bodě 1. může vést k neomezené kumulaci feromonové stopy na hranách jednoho řešení a tím eventuálně i k ustálení na neoptimálním řešení,

platí tedy podmínka pro aktualizaci hodnot feromonové stopy:

$$\tau_{(i,j)} = \left[\tau_{(i,j)} + \Delta \tau^{bs}_{(i,j)} \right]_{(\tau_{min})}^{(\tau_{max})} \tag{12}$$

intenzita feromonových stop je tedy omezena minimální τ^{min} a maximální τ^{max} přípustnou hodnotou. Hodnota $\Delta\tau^{bs}$ – přírůstek intenzity feromonové stopy – je dána rovnicí (10).

3. Dalším rozdílem je vysoká počáteční intenzita feromonu nastavená při inicializaci grafové struktury (viz 3.4 , resp. obr. 14). V případě MMAS se jedná o hodnotu τ^{max} ,která v kombinaci s vysokým koeficientem odpařování ρ umožňuje, aby algoritmus v počátečních iteracích více prozkoumával stavový prostor.
4. Při běhu algoritmu je sledována kvalita řešení, a v případě stagnace dojde k reinicializaci intenzit feromonových stop na všech hranách grafu. Stejně jako dodatečné posílení feromonové stopy v § 3.5.2 je tato reinicializace příkladem využití procedury *deamonActions* z § 3.4 .

Jak je patrné z § 3.5.1 , § 3.5.2 a § 3.5.3 , rozdíly mezi původním algoritmem Ant system a jeho rozšířeními se týkají především rozdílných přístupů k aktualizaci feromonových stop. Odpařování feromonu ani pravidla pro tvorbu řešení se vzájemně nijak neliší.

3.5.5 Ant colony system

Ant colony system se od algoritmu Ant system liší více než algoritmy popsané výše. Podle [10], [12] a [13] se Ant colony system liší od algoritmu Ant system zejména v těchto bodech:

- Agenti používají agresivnější pravidlo pro konstrukci řešení. Pokud se mravenec *k* nachází na vrcholu *i,* pro konstrukci cesty do vrcholu j použije tzv. pseudonáhodné proporční pravidlo dané rovnicí:

$$j = \begin{array}{ll} argmax_{(l \in N_i^k)} \{ \tau_{il} \cdot [\eta_{il}]^{\beta} \} & pokud\ q \leqslant q_0 \\ J & pokud\ q > q_0 \end{array} \tag{13}$$

  kde q je náhodná proměnná s rovnoměrným rozdělením v intervalu $\langle 0\ ,\ 1 \rangle$, q_0 je parametr (přičemž platí $0 \leq q_0 \leq 1$). J představuje cílový vrchol hrany *(i,j)* vybrané na základě rovnice (2), při parametru $\alpha = 1$. N_i^k představuje množinu následníků vrcholu *i,* přičemž tato skupina může být omezena kapacitou, vzdáleností od i, atd. . Nastavením parametru q_0 je možné ovlivnit hledání nových řešení nebo využívání doposud nejlepšího řešení.
- Stejně jako v případě MMAS (viz § 3.5.4) ukládá feromonovou stopu pouze mravenec, který nalezl nejlepší řešení. Na rozdíl od algoritmu Ant system nedochází v tomto algoritmu k odpařování feromonových stop na všech hranách grafu, ale pouze na hranách, které byly použity pro konstrukci nejlepšího řešení. Feromonová stopa $\tau_{(i,j)}$ na hraně *(i,j)* je ve výsledku aktualizována tímto způsobem:

$$\tau_{(i,j)} = \rho \cdot \tau_{(i,j)} + (1-\rho) \cdot \Delta \tau^{bs}_{(i,j)} \qquad \forall (i,j) \in T^{bs} \tag{14}$$

  kde ρ je koeficient odpařování feromonové stopy, T^{bs} je nejlepší nalezené řešení a $\Delta\tau^{bs}$ – přírůstek intenzity feromonové stopy, je dán rovnicí (10). Jak je z rovnice (14) patrné, množství přidávaného feromonu je násobeno koeficientem (1- ρ). Tato vlastnost omezuje kumulaci feromonových stop na hranách nejlepšího řešení a napomáhá prozkoumávání grafové struktury. Globální aktualizace feromonové stopy se uskutečňuje pouze na hranách nejlepšího řešení, což vede ke snížení výpočetní náročnosti algoritmu.
- Aktualizace intenzity feromonových stop v tomto algoritmu je vícestupňová, výše je popsaná globální aktualizace feromonové stopy uskutečňovaná po každé iteraci. Mimo to probíhá za běhu algoritmu ještě lokální aktualizace intenzity feromonových stop a to následujícím způsobem:

$$\tau_{(i,j)} = (1-\xi)\tau_{(i,j)} + \xi \tau_0 \tag{15}$$

  kde ξ je koeficient $0 \leq \xi \leq 1$, τ_0 je hodnota použitá při inicializaci feromonových stop. Tato lokální aktualizace je provedena vždy, když některý z mravenců použije hranu

(i,j) pro konstrukci řešení. Význam této lokální aktualizace spočívá v tom, že každým použitím hrany (i,j) se snižuje intenzita feromonové stopy $\tau_{(i,j)}$ a tím se zároveň snižuje pravděpodobnost, že tato hrana bude použita dalšími mravenci pro konstrukci řešení, čímž se zároveň zvyšuje prohledávání stavového prostoru.

3.6 Problémy řešené pomocí ACO

Jak uvádí [10] a [12], algoritmy ze skupiny ACO algoritmů byly původně použity pro řešení problému obchodního cestujícího, respektive pro řešení problémů spojených s optimálním průchodem sítí. S rozvojem a úpravou původních algoritmů došlo k rapidnímu nárůstu řešených problémů. Algoritmy ze skupiny ACO byly použity pro řešení problémů z těchto oblastí: plánování trasy, přiřazovací problémy, rozvrhovací problémy, toky sítí, barvení grafu, skládání proteinů, strojové učení a množství dalších. Výčet řešených problémů je uveden v literatuře [10] a [12].

Algoritmy ze skupiny optimalizace pomocí mravenčích kolonií byly také použity pro řešení spojitých optimalizačních problémů. Více se této problematice věnuje [13] a [16].

Mravenčí algoritmy jsou také úspěšně používány pro optimalizaci reálných přepravních a distribučních procesů. Přehled společností, které komerčně využívají tyto algoritmy nabízí, práce [16].

3.7 Plánování cesty robota pomocí mravenčích algoritmů

Algoritmy pro plánování cesty robota pomocí ACO vychází z ACO algoritmů použitých pro řešení problému obchodního cestujícího. V obou případech je nutné převést pracovní prostor robota na grafovou strukturu, kde hrany představují možné trajektorie robota. Podle práce [6] je nezbytné v takto popsaném pracovním prostoru určit startovní a cílový bod a tyto body přiřadit jednotlivým mravencům. Literatura uvádí více přístupů k plánování cesty robota pomocí mravenčích kolonií, přičemž tyto přístupy se nepatrně liší od algoritmů řešících problém obchodního cestujícího.

Jak je uvedeno v literatuře [2], [3], [6], v pseudokódu lze algoritmus pro plánování cesty robota popsat takto:

1. Inicializace:
   Vytvořit grafovou strukturu, urči startovní a cílový bod
   Nastav: $t = 0$ (t je časovač)
   $NC = 0$ (NC je čítač cyklů)
   Pro všechny hrany grafu (i,j) nastavit počáteční intenzitu feromonové stopy $\tau(i,j) = C$
   Umístit m mravenců na startovní bod
   for *k:=1* to *m* do{
   Inicializovat paměť M^k *k*-tého mravence
   }
2. repeat{
   for *k:*=1 to *m* do {
   Vybrat bod *j* dle rovnice (2) nebo (13).
   Přesuň mravence *k* do bodu *j*.
   Nastavit $cl_k = cl_k + d_{ij}$ (cl_k je délka cesty *k.* mravence)
   (d_{ij} je délka hrany použité pro cestu do j)

   if($cl_k > cl_{MAX}$) then
   Zrušit k. mravence
   else
   Umístit bod *j* do $Mk^{(s)}$
   }
   }until(dosažen cíl)

3. for *k:=1* to *m* do {
   Umísti *m.* mravence na startovní bod
   Spočítej délku cesty *k.* mravence L_k , urči nejkratší nalezenou cestu
   }

Aktualizovat feromonové stopy na hranách grafu v závislosti na použitém algoritmu
Nastav: $t = t+n$
$NC = NC+1$
(Doplňkové akce – lokální hledání aj.)

4. if (*not podmínky ukončení běhu algoritmu*) then
   Vymazat paměť M^k všem mravencům
   Pokračovat krokem 2.
   else
   Zobrazit nejkratší nalezenou cestu.
   Konec hledání.

Parametr $cl_{MAX,}$ – maximální přípustná délka cesty – má význam omezující podmínky. Tento parametr může být nastaven až při běhu algoritmu a jeho cílem je omezení prohledávání grafové struktury. Použití tohoto parametru může být následující: parametr cl_{MAX} je nastaven až po nalezení prvního řešení a hodnota je následující:

$$cl_{max} = e\, L^1 \tag{16}$$

kde L^1 je délka prvního nalezeného řešení, e je nezáporný koeficient.

Práce [4] uvádí úpravu algoritmu pro hledání cesty robota spočívající v tom, že paralelně existuje více kolonií, které si cyklicky předávají informace o nalezeném řešení. Všechny kolonie řeší jeden konkrétní problém, přičemž použití více kolonií s cyklickou výměnou informací vede podle [4] ke kvalitnějšímu řešení.

Literatura [3] uvádí možnost řešit problém plánování cesty více robotů algoritmem ant colony system, přičemž žádný z robotů nemá shodné startovní a cílové body a plánování cesty jednotlivých robotů probíhá odděleně. Problematika řešení kolizí robotů je naznačena v kapitole 2.4 .

3.8 Nastavení parametrů ACO algoritmů

Jak je uvedeno výše, algoritmy ze skupiny ACO obsahují množství parametrů, které mohou dle [10] velmi výrazně ovlivnit chování algoritmů a kvalitu řešení.

Jedním z důležitých parametrů je velikost mravenčí kolonie. V případě velkého počtu agentů se může zvýšit výpočetní náročnost algoritmu, zatímco při nízkém počtu mravenců se může prodloužit doba potřebná k nalezení řešení o požadované kvalitě, eventuálně požadované kvality řešení nemusí být vůbec dosaženo. Vlivem počtu umělých mravenců na kvalitu řešení se detailněji zabývají práce [1] a [10].

V podkapitole 3.5.1 , resp. v rovnici (2) jsou další dva neméně důležité parametry určující kvalitu řešení. Koeficienty α, β určují důležitost feromonové stopy, resp. vzdálenosti vrcholů grafu na kvalitu řešení. Jak uvádí [10], pro hodnoty $\alpha >> \beta$ dochází k velmi rychlému ustálení pohybu mravenců na jednom řešení, což je nežádoucí kvůli možnému uvíznutí v lokálním optimu. Naopak pro hodnoty $\beta >> \alpha$ dochází při konstrukci řešení k upřednostňování vrcholů s menší vzájemnou vzdáleností – algoritmus se tedy začne chovat jako tzv. hladový algoritmus, což může opět vést k řešení o nižší kvalitě.

Dalším z parametrů výrazně ovlivňujících kvalitu řešení je koeficient odpařování feromonové stopy ρ (viz (4), 3.5.1), pokud je hodnota ρ zvolena nevhodně, může docházet buď k rychlému odpařování feromonových stop a tedy rychlejšímu „zapomínání“ řešení, případně k opačnému chování, kterým je rychlé ustálení pohybu mravenčí kolonie po jednom řešení a jak již bylo zmíněno výše, k možnosti uvíznutí v lokálním optimu.

Další z parametrů, který může výrazně ovlivnit chování ACO algoritmu, je počáteční hodnota feromonové stopy τ_0. Pokud je hodnota τ_0 vysoká, v počátečních iteracích dochází podle [10] k zvýšenému prohledávání grafové struktury, což se může ovlivnit časovou náročnost běhu algoritmu. Naopak pokud je hodnota τ_0 příliš nízká, může dojít k již několikráte zmiňovanému ustálení na lokálním optimu.

Algoritmy popsané v § 3.5.2 , 3.5.3 , 3.5.4 a 3.5.5 obsahují další parametry výrazně ovlivňující řešení. Optimální nastavení těchto parametrů, stejně tak parametrů zmíněných výše, je podle [10] mnohdy klíčem k požadovanému fungování algoritmů.

3.8.1 Podmínky ukončení běhu algoritmu

Jak vyplývá z kapitol 3.4 a 3.7 , běh ACO algoritmu je přerušen při dosažení určitých ukončujících podmínek. Podmínek ukončení běhu algoritmu může být více a je možné je kombinovat [15]. Nejčastěji používanými podmínkami jsou:

- časové kritérium,
- maximální počet iterací,
- nalezení dostatečného počtu řešení,
- ustálení řešení v určité oblasti.

3.9 Srovnání ACO a genetických algoritmů

Práce [17] a [18] uvádí srovnání algoritmů ze skupiny optimalizace pomocí mravenčích kolonií a genetických algoritmů. Jak zmíněné práce uvádí, genetické algoritmy jsou vhodnější pro hledání optimálního řešení, zatímco ACO algoritmy se častěji pohybují v suboptimální oblasti (jak také uvádí [10]). Podle prací [17],[18] je použití ACO algoritmu vhodnější v případě prostředí s velkým počtem překážek, vyšší počet překážek nezvyšuje algoritmům ze skupiny ACO, na rozdíl od genetických algoritmů, nároky na zdroje. Literatura dále uvádí, že vhodnou kombinací ACO algoritmů a genetických algoritmů by bylo možné dosáhnout lepších výsledků v kratším čase. Navrhované řešení spočívá v globálním hledání cesty pomocí mravenčích algoritmů a doplnění lokálního prohledávání pomocí genetického algoritmu. Více se této problematice věnuje práce [6], kde je uvedena tato kombinace algoritmů.

4 NÁVRH ALGORITMŮ PRO PLÁNOVÁNÍ CESTY ROBOTA

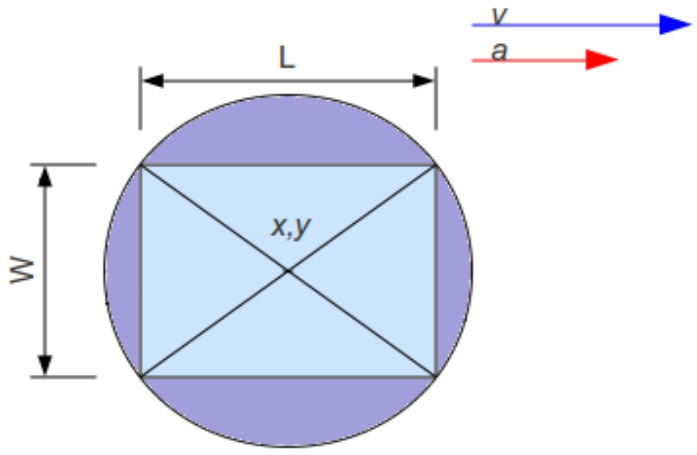
Navrhovaný algoritmus řeší problém plánování cesty robota daným prostředím. Algoritmus je vícestupňový, nejprve proběhne předzpracování, kdy je pracovní prostor robota převeden na konfigurační prostor C. Volný konfigurační prostor $C_{free} \subset C$ je popsán grafem a ve druhé etapě běhu algoritmu probíhá vlastní hledání cesty. Po nalezení cesty o požadovaných parametrech, respektive po ukončení hledání, je na nalezené řešení aplikovaná procedura vyhlazení cesty a takto upravená cesta je převedena na trajektorii, po které se posléze pohybuje daný robot.

Metody pro plánování cesty vychází z práce [15], kterou ovšem v několika směrech rozšiřují. Jedná se především o použití více algoritmů pro plánování cesty a možnost pracovat s komplikovanějšími překážkami. Dalším rozšířením je také mechanismus vyhlazení trajektorie popsaný v kapitole 4.6 a možnost plánování cesty více robotů současně.

4.1 Robot

Uvažovaným robotem pro plánování cesty bude všesměrový holonomní robot, který je dán svými rozměry $L \times W$, pro zjednodušení nahradím robota kružnicí o poloměru $R = 0{,}5 \cdot \sqrt{W^2 + L^2}$ se středem v geometrickému středu robota. Dále je robot specifikován maximální dopřednou rychlostí v_{max} a zrychlením a.

Robot se v daném pracovním prostředí začíná pohybovat v čase t_0, počáteční rychlost robota je nulová a robot se pohybuje z počáteční pozice $S(x_S, y_S)$ na cílové pozice $C = \{C_1, \ldots C_n\}, C_n \geq 1$, kde cílová pozice Ci je dána souřadnicemi x^C_i, y^C_i. Pokud má robot zadáno více cílových bodů, je možné nastavit pořadí v jakém mají být cílové body navštíveny.



Obr. 15 Uvažovaný robot

4.1.1 Pohyb robota

Robot se pohybuje v prostoru s polygonálními překážkami po nekolizní trajektorii vytvořené plánovacím algoritmem. Předpokládám, že robot se od vypočtené trajektorie neodchyluje a během pohybu robota tedy neprovádím detekci kolizí s překážkami.

Jak je uvedeno výše, robot je uvažovaný jako holonomní všesměrový. Pohybuje se po naplánované cestě tak, aby dosáhl splnění cílů a to v pokud možno takovým způsobem, který minimalizuje ujetou vzdálenost nebo časovou náročnost přesunu ze startu do cíle.

4.2 Pracovní prostor robota

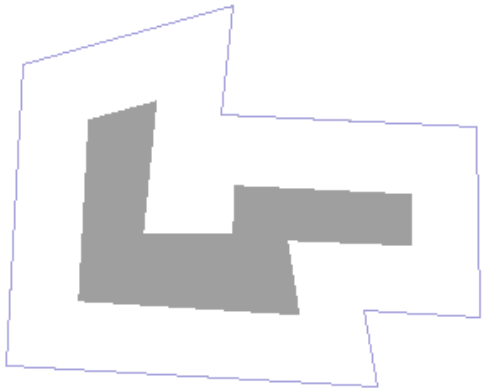
Pracovní prostor robotů je uvažován jako diskrétní dvourozměrný prostor omezený svými hranicemi. Pracovní prostor robota je uvažován jako statický s polygonálními překážkami. Překážky mohou být konvexní i konkávní, nepřípustné jsou sebe-protínající polygony (v literatuře [23] označované jako komplexní polygony).

Obr. 16 Pracovní prostor robota s překážkami

4.2.1 Konfigurační prostor a předzpracování

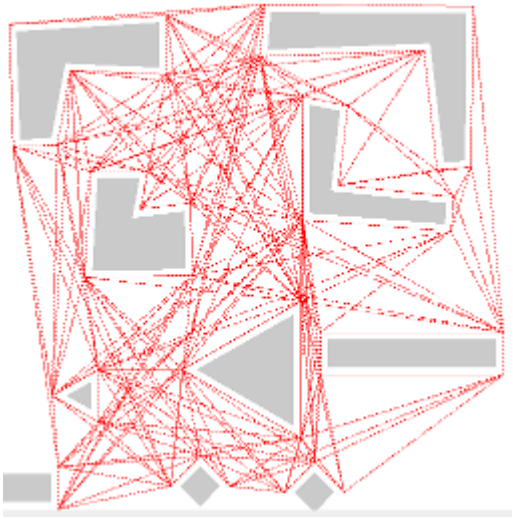
Všechny překážky v pracovním prostoru robotů jsou před vlastním hledáním cesty expandovány o parametr *R* z § 4.1 , jak naznačuje obr. 17 . Sjednocení všech expandovaných překážek vytváří kolizní konfigurační prostor C_{obs} , rozdíl pracovního prostoru a kolizního konfiguračního prostoru tvoří volný konfigurační prostor C_{free}. Vrcholy expandovaných překážek, které neleží uvnitř jiných expandovaných překážek, jsou dále použity jako vrcholy grafu popisujícího volný konfigurační prostor. Spojnice vrcholů neprotínající žádnou z překážek jsou použity jako hrany grafu, pro popis volného konfiguračního prostoru byl tedy zvolen graf viditelnosti, jak je patrné z obr. 18 .

Výše popsaná expanze překážek umožňuje redukci popsaného v kapitole 4.1 na bod, a ve výsledku expanze překážek umožňuje redukci konfiguračního prostoru a zjednodušení plánování cesty.



Obr. 17 Expanze překážky

Jak je uvedeno výše, plánování cesty probíhá ve více krocích. V rámci předzpracování jsou expandovány překážky a z expandovaných překážek je vytvořen orientovaný graf viditelnosti (viz obr. 17 a obr. 18). V rámci předzpracování jsou dále do grafu popisujícího volný konfigurační prostor přidány startovní a cílové body robota. Hrany grafu představují možné cesty, kudy se může robot uvedený v podkapitole 4.1 pohybovat bez rizika kolize s překážkami. Vrcholy grafu představují místa, kde robot mění směr pohybu. Cílem algoritmů pro plánování cesty popsaných dále je nalezení cesty o minimální délce, případně cesty o délce blízko minimu.



Obr. 18 Graf viditelnosti popisující C_{free}

4.3 Mravenčí kolonie

Pro plánování cesty uvedeného robota je použita kolonie *m* umělých mravenců. Mravenci mají přiděleny startovní a cílové body, přičemž rozpoznávají dosažení cílového bodu. Každý z mravenců má paměť M^k, do které ukládá komponenty grafu použité pro konstrukci řešení. Tato paměť zajišťuje, aby mravenec nepoužil pro konstrukci řešení jednu komponentu grafu dvakrát. V paměti mravence jsou také zaznamenané informace o délce cesty a počtu kroků.

Mravenci se pohybují po hranách grafu vytvořeného během předzpracování, čímž konstruují řešení. Pohyb je pravděpodobnostní a je ovlivňován intenzitou feromonové stopy na hranách grafu a délkou těchto hran. Pravděpodobnost, že se mravenec nacházející se na vrcholu *i* rozhodne pro přesun do vrcholu *j,* je dána následující rovnicí:

$$p_{(j)}^{k} = \begin{cases} \dfrac{|\tau_{(i,j)}|^{\alpha} \cdot |\eta_{(i,j)}|^{\beta}}{\sum_{z \notin M^k} |\tau_{(i,z)}|^{\alpha} \cdot |\eta_{(i,z)}|^{\beta}} & \text{pokud } j \notin M^k \\ 0 & \text{pokud } j \in M^k \end{cases} \tag{17}$$

,kde $\tau_{(i,j)}$ představuje intenzitu feromonové stopy na hraně *i,j,* $\eta_{(i,j)}$ představuje heuristickou informaci spojenou s hranou *(i,j)* danou převrácenou hodnotou délky hrany *(i,j)*. Proměnná *z* představuje všechny bezprostřední sousedy vrcholu *i*. Exponent α určuje důležitost hodnoty feromonové stopy při výběru další cesty. Naopak exponent β určuje důležitost vzdálenosti při výběru další cesty. Rovnice (17) je použita pro hledání cesty ve všech dále uvedených algoritmech, ovšem pro případ algoritmu Ant colony system je doplněna o následující pravidlo výběru cesty:

$$j = \begin{cases} argmax_{(l \in N_i^k)} \{ [\tau_{(i,j)}]^{\alpha} \cdot [\eta_{il}]^{\beta} \} & pokud\ q \leqslant q_0 \\ J & pokud\ q > q_0 \end{cases} \tag{18}$$

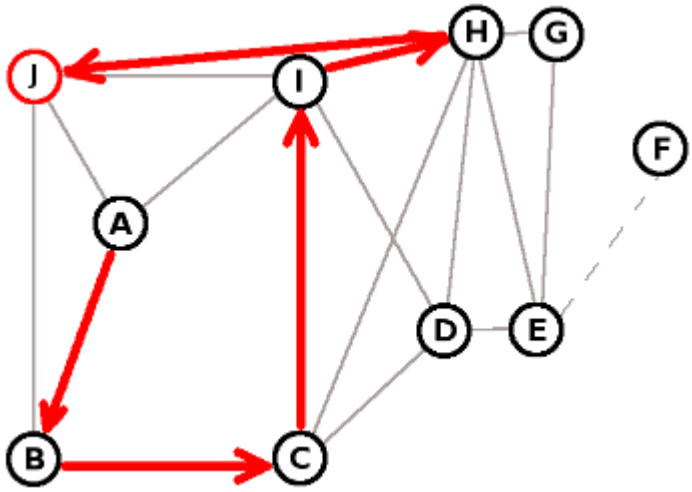
kde *q* je náhodná proměnná s rovnoměrným rozdělením v intervalu $\langle 0\ ,\ 1 \rangle$, q_0 je parametr a platí $q_0 = \langle 0\ ,\ 1 \rangle$. *J* představuje cílový vrchol hrany *(i,j)* vybrané na základě rovnice (17), N_i^k představuje množinu následníků vrcholu *i,* takových, že žádný z vrcholů obsažených v N_i^k není uložen v paměti M^k. Algoritmus Ant colny system tedy v případě, že $q_0 > q$ volí pro pokračování konstrukce řešení takový vrchol, do něhož vede hrana s nejvyšší kombinací heuristické informace a intenzity feromonové stopy.

Mravenci mají také v závislosti na použitém algoritmu plánování cesty schopnost ukládat na hrany grafu umělé feromonové stopy. Pravidla použitá pro ukládání feromonových stop jsou uvedena níže v § 4.4.2 – 4.4.6 .

Stejně jako v literatuře [3] a [15], tak i zde je v návrhu umělých mravenců uvažována procedura, která mravenci vymaže paměť a umístí jej zpět na startovní pozici. Ekvivalentem může být

eliminace mravence a vytvoření nového na startovní pozici. Význam má tato procedura především v situaci, kterou naznačuje obr. 19 . Mravenec se po 5 krocích ocitl ve vrcholu, odkud už nemůže nikam pokračovat, protože všechny okolní vrcholy již navštívil. Pokud by nebyla procedura vymazání paměti použita, mravenec by se nadále nepodílel na konstrukci řešení. Vzhledem k tomu, že obdobná situace může hypoteticky postupně nastat u všech členů kolonie, mohlo by dojít k uváznutí algoritmu. Z toho důvodu je v návrhu algoritmu nutná procedura restartování mravence.

Částečně omezit riziko uvíznutí mravence je možné po odstranění takových vrcholů z grafové struktury, které mají jen jednoho následníka a zároveň nepatří do množiny startovních a cílových bodů. Tyto vrcholy a hrany s nimi spojené nemohou být použity pro konstrukci řešení a není nutné je v grafu uchovávat.



Obr. 19 Reset mravence po 5 krocích, odstranění vrcholu

4.4 Návrh algoritmů pro plánování cesty

Vlastní plánování cesty robota, tedy hledání optimální posloupnosti hran a vrcholů spojující startovní a cílové body, je realizováno kolonií umělých mravenců. Algoritmy pro hledání cesty vychází z algoritmů popsaných v § 3.5.1 – 3.5.5 . Vzhledem k velkému počtu shodných částí v těchto algoritmech je vhodné realizovat výchozí konstrukci algoritmu a provést rozšíření pro jednotlivé algoritmy, při implementaci algoritmů je pak možné použít výhod abstraktních tříd. Vzhledem k tomu, že všechny algoritmy pro hledání cesty použité v této práci vychází z algoritmu Ant system popsaného v § 3.5.1 , je vhodné použít jako základní konstrukci právě tento algoritmus.

4.4.1 Podmínky ukončení hledání cesty

Při hledání cesty je nutné stanovit podmínky, při kterých dojde k ukončení hledání. Pokud by tomu tak nebylo, algoritmus by proces hledání cesty nikdy neukončil.

Pro všechny algoritmy navrhované dále předpokládám stejné podmínky ukončení hledání cesty. V návrhu algoritmu předpokládám následující podmínky ukončení hledání cesty.

- Dosažení časového limitu běhu algoritmu.
- Nalezení stanoveného počtu cest (včetně případných duplicit).
- Dosažení stanoveného počtu cyklů hledání.
- Ustálení výsledků v určitém rozmezí hodnot – stagnace.

Rozdíl mezi cyklem hledání a nalezením cesty spočívá v tom, že v rámci jednoho cyklu je nalezena minimálně jedna cesta. Kontrola stagnace spočívá v tom, že je stanoveno minimální přípustné zlepšení výsledku na určitý počet cyklů. Pokud je zlepšení menší než minimální přípustné, běh algoritmu je ukončen.

Pro ukončení hledání stačí, aby byla dosažena jediná z uvedených podmínek.

4.4.2 Základní konstrukce algoritmu hledání cesty

Jak je uvedeno výše, základní konstrukce algoritmu pro hledání cesty používá algoritmus Ant

system. Algoritmus je možné rozčlenit na dvě dílčí části, a to inicializaci, a dále pak vlastní hledání cesty s určením optimální cesty. Výsledkem plánování cesty je posloupnost hran a vrcholů spojující počáteční pozici s cílovými body, tento výsledek je po ukončení plánování předán k dalšímu zpracování – viz kapitola 4.6 .

V pseudokódu uvádím návrh algoritmu pro plánování cesty robota pomocí algoritmu Ant system, předpokládám existenci scény s polygonálními překážkami, ve které jsou určené startovní a cílové body robota. Předpokládám, že startovní a cílové pozice nekolidují s překážkami.

1. **Inicializace:**

   Předzpracování:

   → Expanze překážek.

   → Vytvoření grafu viditelnosti.

   → Připojení startovních a cílových bodů do grafu.

   → Odstranění vrcholů s jedním následníkem a hran mezi těmito vrcholy z grafové struktury.

   Nastavit mravencům parametry α, β.

   Nastavit parametr ρ.

   CC := 0. (CC je čítač cyklů)

   GR := 0. (GR je čítač dosažení cíle)

   $L_{best} := \infty$. (L_{best} je délka nejlepšího nalezeného řešení)

   Na všech hranách *(i,j)* nastavit počáteční hodnotu umělého feromonu $\tau^{start}_{(i,j)}$.

2. **Hledání cesty:**

   Vytvořit kolonii *m* mravenců, umístit všechny mravence do startovního bodu.

   Pro všechny mravence platí: aktuální pozice = startovní bod.

   Inicializovat paměť M^k všem mravencům, ulož aktuální pozici do M^k.

   Nastavit všem mravencům cílové body podle robota, pro kterého je plánovaná cesta.

   $L_{max} := \infty$. (L_{max} je maximální přípustná délka cesty)

   Nastav: $x = m$.

   repeat {

   for(k := 1 to m) do {

   if(mravenec *k* již ukončil konstrukci řešení) then

   {continue.}

   if (mravenec *k* uvízl) then

   {reset *k*. mravence.}

   Přesuň mravence k. do bodu *j* vybraného podle rovnice (17).

   Aktuální pozice = *j*, přidej *j* do M^k.

   Nastav: $C_l^k = C_l^k + d_{ij}$. (C_l^k je délka cesty k. mravence,)

   (d_{ij} je délka hrany mezi *i* a *j*)

   if($C_l^k > L_{max}$) {

   Nastav: *k*. mravenec neúspěšně ukončil konstrukci řešení.

   (nenalezeno řešení, *k*. mravenec se dál nebude podílet na konstrukci řešení)

   x := x – 1.

   continue.}

   if(cílové body k. mravence obsahuji aktuální pozici.) {

   Odstraň aktuální pozici z cílových bodu k. mravence.

if(cílové body k. mravence prázdné) {
GR := GR + 1.
x := x – 1.
Nastav: *k*. mravenec úspěšně ukončil konstrukci řešení.
(nalezeno řešení, k. mravenec se dál nebude podílet na konstrukci řešení)
Ulož řešení k. mravence do nalezených řešení .
if($L_{best} > C_l^k$) then {
$L_{best} := C_l^k$.
Nastav řešení mravence *k j*ako nejlepší nalezené řešení .
continue.
} else {
Částečně vymaž paměť k. mravence.
(k. mravenec nalezl jeden z více cílových bodů, hledá tedy dále a při dalším hledání je mu umožněno procházet již navštívené pozice, ale v paměti uchovává částečné řešení, které je zachováno i při případných restartech.)
Nastav: startovní bod k. mravence := aktuální pozice
(při případném restartu nemusí mravenec konstruovat cele řešení znovu, rozšiřuje již nalezené částečné řešení.)
}}}}
} until (x = 0 or dosažen časový limit řešení)

3. Vyhodnocení nalezeného řešení a aktualizace hodnot umělého feromonu:

Nastav *CC:= CC+1*.

Na všechny hrany grafu *(i,j)* s intenzitou feromonové stopy $\tau_{(i,j)}$ aplikuj odpařováni feromonu:

$$\tau_{(i,j)} = \rho \cdot \tau_{(i,j)} \quad \text{, kde } \rho \text{ je koeficient odpařování feromonu, } 0 \leq \rho \leq 1 \tag{19}$$

for(k := 1 to m) {
if(mravenec *k* úspěšně ukončil konstrukci řešení) then {
Urči množství ukládaného feromonu pro *k*. mravence.

$$\Delta \tau_{(i,j)}^k = \frac{1}{C_l^k} \quad \text{, } C_l^k \text{ je délka řešení k. mravence} \tag{20}$$

Nastav na všech hranách *(i,j)* použitých pro konstrukci řešení *k.* mravence novou hodnotu feromonové stopy:

$$\tau_{(i,j)} = \tau_{(i,j)} + \Delta \tau_{(i,j)}^k \tag{21}$$

}}

4. Zpracování výsledků, kontrola dosažení ukončujících podmínek

if(splněny podmínky ukončení hledání z § 4.4.1 *){*
Ulož výsledky hledání.
Předej nejlepší nalezené řešení k dalšímu zpracování.
Konec hledání cesty.
} else {
Pokračuj 2. krokem}

Jak je z výše uvedeného patrné, v návrhu algoritmu připouštím zadání více cílových bodů. Zvolil jsem přístup, kdy v případě více cílů musí mravenci zkonstruovat kompletní řešení přes všechny cílové body. Alternativní přístup by mohl spočívat v tom, že celá kolonie mravenců by se rozdělila na několik podkolonií, a ty by individuálně hledaly jednotlivé cíle, přičemž by platilo, že cílový bod *i*. kolonie je startovním bodem *i+1*. kolonie. Po nalezení všech dílčích řešení by bylo nutné zkompletovat tato dílčí řešení do jednoho celku.

Z výše uvedeného pseudokódu je také patrné použití omezující podmínky – maximální přípustné délky řešení L_{max}. Důvod použití této omezující podmínky je ten, že v případě nelezení prvního řešení v daném cyklu musí každé další řešení být maximálně o stejné délce, každé další řešení nalezené v daném cyklu je tedy lepší než předchozí, event. stejně dobré.

V rámci experimentů bude u tohoto algoritmu zkoumán vliv koeficientů α,β a ρ na kvalitu a rychlost nalezení řešení.

4.4.3 Elitist ant system

Tento algoritmus rozšiřuje výše uvedený Ant system v posilování nejlepšího nalezeného řešení. Dojde k uložení dodatečného množství feromonu na hrany, které byly použity pro konstrukci nejlepšího řešení. Změna algoritmu se tedy týká *1.* a *3.* kroku, které budou doplněny o následující konstrukce:

1. krok*:* Nastavit váhu nejlepšího řešení *e.*

3. krok: Urči přírůstek hodnoty umělého feromonu na hranách nejlepšího řešení:

$$\Delta \tau_{(i,j)}^{bs} = \frac{e}{L_{best}} \quad , L_{best} \text{ je délka nejlepšího řesení} \tag{22}$$

Nastav na všech hranách *(i,j)* použitých pro konstrukci nejlepšího řešení novou hodnotu feromonové stopy:

$$\tau_{(i,j)} = \tau_{(i,j)} + \Delta \tau_{(i,j)}^{bs} \tag{23}$$

Předpokládám, že uvedený koeficient *e* – váha nejlepšího řešení, bude mít klíčovou roli v chování algoritmu, proto budou experimenty s plánováním cesty pomocí tohoto algoritmu zaměřeny především na nastavení tohoto koeficientu. Dále bude stejně jako v předchozím případě zkoumán vliv koeficientů α,β a ρ na kvalitu a rychlost nalezení řešení.

4.4.4 Rank-based ant system

Stejně jako výše uvedený Elitist ant system je i tento algoritmus rozšířením algoritmu Ant system a rozšíření se týká ukládání feromonových stop. Hlavní myšlenka rozšíření původního algoritmu je ta, že jen nejlepších *w* z *m* mravenců v kolonii ukládá feromonovou stopu, přičemž množství ukládaného feromonu je násobeno váhou v rozmezí $\langle 1 , w \rangle$. Dále proběhne stejně jako u Elitist ant system uložení feromonové stopy na hrany nejlepšího nalezeného řešení.

Změna se stejně jako u Elitist ant system týká *1. a 3.* kroku algoritmu Ant system, přičemž úprava *3.* kroku algoritmu nespočívá v doplnění o další konstrukce, ale předefinování celého *3.* kroku.

První krok bude doplněn o následující konstrukci:

Nastavit hodnotu *w.*

Třetí krok má následující tvar:

Vyber *w* nejlepších řešení v dané iteraci.

Nastav *CC:= CC+1*.

Na všechny hrany grafu *(i,j)* s intenzitou feromonové stopy $\tau_{(i,j)}$ aplikuj odpařováni

feromonu podle rovnice(19).

For($i = w$ downto 1) do {

Urči přírůstek hodnoty umělého feromonu na hranách *i.* řešení:

$$\Delta \tau^{i}_{(i,j)} = \frac{i}{L_i} \quad , L_i \text{ je délka } i \text{ . řesení} \tag{24}$$

Nastav na všech hranách *(i,j)* použitých pro konstrukci i. řešení novou hodnotu feromonové stopy:

$$\tau_{(i,j)} = \tau_{(i,j)} + \Delta \tau^{i}_{(i,j)} \tag{25}$$

}

Podle rovnice (22) urči přírůstek hodnoty umělého feromonu na hranách nejlepšího řešení s nastavením $e = w + 1$.

Nastav na všech hranách *(i,j)* použitých pro konstrukci nejlepšího řešení novou hodnotu feromonové stopy –viz (23).

Předpokládám, že v rámci experimentů dosáhne tento algoritmus srovnatelných výsledků jako výše uvedený Elitist AS. Experimenty s tímto algoritmem budou zaměřeny na stanovení takových parametrů *w,* α,β a ρ, které umožní nalezení optimálního řešení.

4.4.5 Max–min ant system

Rozdíl mezi tímto algoritmem a Ant systemem spočívá stejně jako u algoritmů výše v postupu při ukládání feromonové stopy. Na rozdíl od algoritmů uvedených výše ukládá MMAS při běhu feromonovou stopu pouze na hrany nejlepšího nalezeného řešení. Další odlišností je zavedení parametrů τ_{max} a τ_{min}, význam těchto parametrů spočívá se stanovení maximální, resp. minimální přípustné intenzity feromonových stop na hranách grafu. Jak je uvedeno v § 3.5.4 , dalším rozšířením algoritmu MMAS může reinicializace feromonové stopy v případě stagnace. Vzhledem k tomu, že v návrhu algoritmu předpokládám ukončení hledání cesty při stagnujícím chování, tak reinicializaci grafové struktury v návrhu algoritmu MMAS neuvažuji.

První krok algoritmu z § 3.5.1 bude v případě MMAS rozšířen následujícím způsobem:

Nastavit hodnoty τ_{max} a $\tau_{min.}$

Na všech hranách *(i,j)* nastavit počáteční hodnotu umělého feromonu $\tau_{max.}$

Třetí etapa běhu algoritmu bude následující:

Nastav $CC := CC+1$.

Na všechny hrany grafu *(i,j)* s intenzitou feromonové stopy $\tau_{(i,j)}$ aplikuj odpařováni feromonu:

$$\tau_{(i,j)} = max(\tau_{min}, min(\rho \cdot \tau_{(i,j)}, \tau_{max})) \tag{26}$$

Podle rovnice (22) urči přírůstek hodnoty umělého feromonu na hranách nejlepšího řešení s nastavením $e = 1$.

Nastav na všech hranách *(i,j)* použitých pro konstrukci nejlepšího řešení novou hodnotu feromonové stopy.

$$\tau_{(i,j)} = max(\tau_{min}, min(\tau_{(i,j)} + \Delta \tau^{bs}, \tau_{max})) \tag{27}$$

$$\text{Nastav} \quad \tau_{max} = \frac{1}{(L_{best} \cdot (1-\rho))}, \tau_{min} = \frac{\tau_{max}}{5} \tag{28}$$

V rámci experimentů bude zkoumán především vliv parametrů τ_{max} a τ_{min} na kvalitu řešení. Na základě informací z literatury [10] předpokládám, že algoritmus MMAS bude pro kratší časové úseky dosahovat přibližně stejných výsledků hledání cesty jako výše uvedené algoritmy, ale pro delší dobu běhu předpokládám lepší výsledky, především pak rychlejší nalezení optimální cesty, eventuálně cesty blízko optimu.

4.4.6 Ant colony system

Jak je uvedeno v kapitole 3.5.5 , Ant colony system se od algoritmu Ant system liší více než algoritmy uvedené výše. Hlavní rozdíl spočívá v pravidle pro konstrukci řešení – viz (18), dalším rozdílem je přístup k aktualizaci hodnot feromonových stop.

První etapa algoritmu Ant colony system je doplněna takto:

Nastav parametr q_0.

Nastav parametr α=1.

Nastav parametr ξ.

Druhá etapa tohoto algoritmu se od algoritmu Ant system liší odlišným pravidlem pro konstrukci řešení a lokální aktualizací feromonové stopy.

Přesuň mravence k. do bodu *j* vybraného podle rovnice (18).

Proveď lokální aktualizaci feromonové stopy na poslední použité hraně *(i,j):*

$$\tau_{(i,j)} = (1-\xi)\cdot\tau_{(i,j)} + \xi\cdot\tau_{(i,j)}^{start} \qquad (29)$$

Ukládání feromonových stop po dokončení iterace je v tomto algoritmu realizováno obdobně jako ve výše uvedeném Max–min ant system, tedy pouze na hranách, které byly použity pro konstrukci nejlepšího řešení, dochází k aktualizaci feromonových stop. Rozdíl mezi tímto algoritmem a ostatními výše zmíněnými spočívá v mechanismu odpařování feromonových stop. Ant system a jeho varianty (§ 4.4.2 – 4.4.5) provádí odpařování feromonu na všech hranách grafové struktury, Ant colony system provádí odpařování pouze na hranách použitých pro konstrukci nejlepšího řešení. Třetí etapa běhu algoritmu bude mít následující tvar:

Nastav *CC:= CC+1*.

Podle rovnice (22) urči přírůstek hodnoty umělého feromonu na hranách nejlepšího řešení s nastavením *e = 1*.

Nastav na všech hranách *(i,j)* použitých pro konstrukci nejlepšího řešení novou hodnotu feromonové stopy

$$\tau_{(i,j)} = \rho\cdot\tau_{(i,j)} + (1-\rho)\cdot\Delta\tau^{bs} \qquad (30)$$

V rámci experimentů s tímto algoritmem bude zkoumán vliv parametrů ξ, q_0,aj. na kvalitu a rychlost řešení. Předpokládám, že tento algoritmus bude dosahovat přibližně stejných výsledků jako Max min ant system.

4.4.7 Lokální prohledávání

Výše uvedené algoritmy konstruují pomocí kolonie umělých mravenců pohybujících se po hranách grafu řešení spojující startovní a cílové body. Vzhledem k tomu, že prohledávání grafu probíhá postupně, je tedy časově závislé, není možné zaručit, že nalezené řešení bude optimální. Předpokládám, že tento problém bude patrný zejména tehdy, pokud bude nastaven nízký časový limit pro hledání cesty daným prostředím. Pro řešení této situace je vhodné do algoritmů zakomponovat funkci lokálního prohledávání. Jedná se o příklad užití funkce *deamonActions* uvedené v kapitole 3.4 .

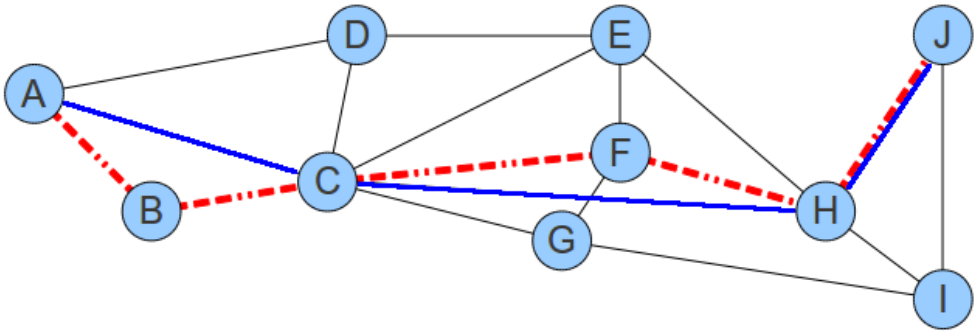
Funkce lokálního prohledávání, respektive úpravy nalezeného řešení bude zakomponovaná do

druhé etapy běhu algoritmu. Funkce bude volitelná, tedy bude možné ji při běhu algoritmu nevyužít. V pseudokódu uvádím návrh této funkce. Funkce bude mít jeden parametr, a to řešení nalezené některým z plánovacích algoritmů – sekvenci hran. V případě hledání cesty s více cílovými body bude funkce volána vždy pro úpravu částečného řešení, nebude možné ji použít na vylepšení takového řešení, kde budou některé z cílových bodu na jiné než koncové či počáteční pozici.

```
function vylepšeni_cesty ( parametr: posloupnost_hran) {
1.      Inicializuj navratova_hodnota.
2.      Nastav i := 1.
3.      while ( i < (posloupnost_hran → délka) ) do
        {
4.      Aktuální vrchol :=  i. hrana → výchozí vrchol hrany
5.      for( j = (posloupnost_hran → délka )- 1 downto i ) do
        {
6.      Vrchol2 = j. hrana → cílový vrchol hrany.
7.      Hledej v hranách vedoucích z Aktuální vrchol takovou, která končí ve Vrchol2.
8.      if(hrana nalezena) then
        {
9.      Přidej nalezenou hranu do navratova_hodnota.
10.     Nastav i:= j.
11.     Pokračuj na krok 3.
        }}
12.     Přidej i. hranu do navratova_hodnota.
13.     I := i +1.
        }
14.     return navratova_hodnota. }
```

Jak je z návrhu patrné, jedná se slepé prohledávání využívající trojúhelníkové nerovnosti. Pro pochopení toho, jak pracuje tato funkce, poslouží obr. 20 . Červeně je vyznačeno řešení, které je třeba upravit, modře výsledek po úpravě. Vzdálenosti na obrázku odpovídají délce hran. Z trojúhelníkové nerovnosti plyne, že délka hrany A-C je menší než součet délek hran A-B a B-C. Proto je pro konstrukci řešení použita hrana A-C. Stejný postup je aplikován na hrany C-H, H-F, F-C. Hranu H-J není možné nahradit jinou – kratší hranou, a proto je použita pro konstrukci řešení.

V rámci experimentů bude zkoumáno, za jakých podmínek bude výhodné aplikovat tuto funkci na úpravu řešení a kdy naopak nebude nutné aplikovat, a to buď z důvodu nalezení dostatečně kvalitního řešení, nebo snížení časové náročnosti algoritmu.

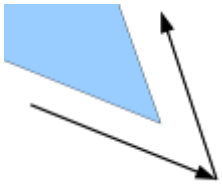


Obr. 20 Úprava nalezeného řešení

4.5 Vyhlazování cesty

Předpokládejme, že uvažovaný holonomní všesměrový robot se může pohybovat po cestě vytvořené z hran grafu popisujícího volný stavový prostor. Při pohybu přímo po hranách grafu pak může nastat situace, kdy pro pokračování pohybu musí robot nejprve zastavit, a poté se opět rozjet–

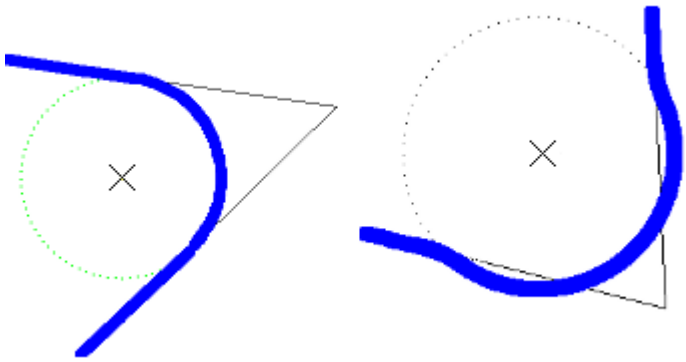
jak naznačuje obr. 21 . Tato situace je způsobena tím, že velikost vektoru zrychlení v daném okamžiku je větší, než hodnota zrychlení daného robota z § 4.1 . V takové situaci dochází k časovým, případně i energetickým ztrátám, a bylo by tedy vhodné cestu upravit tak, aby robot nemusel během pohybu mezi startem a cílem zastavovat. Úprava naplánované cesty robota může být výhodná nejen při situaci naznačené obr. 21 , ale kdykoli, kdy úpravou cesty dojde k jejímu zkrácení nebo dojde ke zkrácení doby potřebné pro dokončení úkolu.



Obr. 21 Cesta robota

Řešení problému úpravy cesty nabízí práce [8], a to takové, že hrany, po kterých se pohybuje robot, jsou proloženy hladkou křivkou – kruhovým obloukem, jak naznačuje obr. 22 -vlevo. Výhodou tohoto řešení je kromě jiného zkrácení délky cesty a zkrácení času potřebného pro přesun ze startu do cíle. Nevýhodou tohoto přístupu je fakt, že při takovémto vyhlazování cesty je nutné kontrolovat, zda se robot během pohybu nedostane do nepřípustné konfigurace. Pokud taková situace nastane, jsou možná dvě řešení:

- Vyhlazování cesty robota nebude aplikováno a robot se tedy bude pohybovat přímo po hranách grafu s tím, že dle potřeby zastaví a rozjede se jiným směrem.
- Vyhlazování cesty robota použije jiný přístup, jak je naznačeno na obr. 22 vpravo, kde byly pro vyhlazení cesty použity tři navazující kruhové oblouky.



Obr. 22 Vyhlazení cesty

V algoritmu plánování cesty tedy předpokládám funkci, která provede vyhlazení cesty robota. Tato funkce provede vyhlazování cesty tak, že cesta robota bude kratší nebo dojde k úspoře času, případně funkce vyhodnotí, že vyhlazování cesty je nerealizovatelné či nevýhodné. V pseudokódu uvádím návrh funkce vyhlazování cesty robota. Parametrem funkce je posloupnost hran – výsledek plánování cesty grafem pomocí výše uvedených algoritmů.

function vyhlazeni_cesty (parametr: posloupnost_hran)

1. for(*i* = *1* to ((posloupnost_hran → délka) - 1)) do
   {
2. Hrana A = posloupnost_hran → *i*. hrana
3. Hrana B = posloupnost_hran → *i+1*.hrana
4. Nastav poloměr R_z = parametr R z § 4.1
5. Na ose úhlu najdi takový bod S_z, že | S_z A| = | S_z B| = R z § 4.1 .
6. Zkonstruuj konvexní kruhový oblouk se středem v S_z o poloměru R takový, že hrany A, B jsou jeho tečny a zároveň body dotyku s hranami A,B jsou krajní body kruhového oblouku.
7. Proveď kontrolu přípustné konfigurace po celé délce kruhového oblouku.
8. if(vyhovuje)then {

do{

9. Zvětší poloměr R.
10. Opakuj postup podle 5. – 7.}
11. while(vyhovuje podmínce 7.)}
12. Nahraď příslušnou část hran A, B vytvořeným kruhovým obloukem .
13. continue.

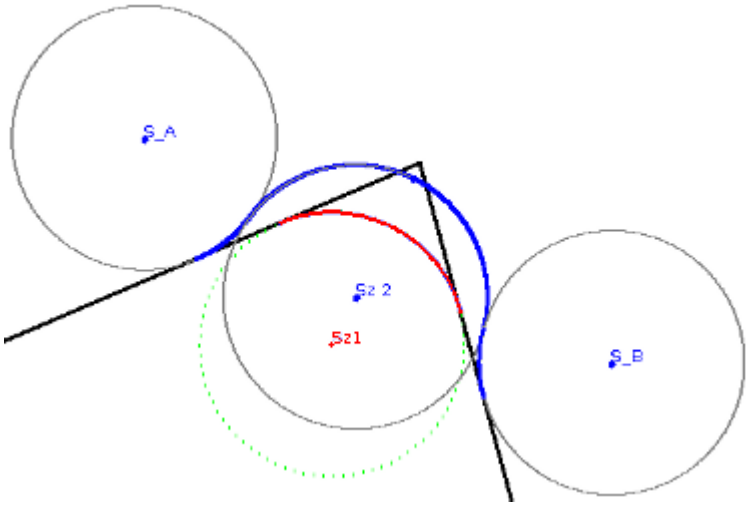
}else {

repeat{

14. Posuň S_z v ose úhlu blíže průsečíku hran A a B.
15. Zkonstruuj tři kružnice o poloměrech R se středy v S_A, S_B,S_Z
    Platí: $|S_A S_z| = 2$ R, $|S_A A| = R$, $|S_B S_z| = 2$ R, $|S_B B| = R$, $|S_A S_B| > 2R$
16. Z konvexních kruhových oblouků vzniklých z kružnic uvedených v 15.zkonstruuj alternativní cestu
17. Proveď kontrolu konfigurací robota na této cestě
18. }until (vyhovuje or (alternativní cesta je delší a pomalejší než cesta po hranách A,B))
19. if(Vyhovuje podmínce 17.) nahraď příslušnou část hran A, B alternativní cestou.
    }}
20. return upravena_posloupnost_hran.

Jak je výše uvedeno, po celé délce alternativní cesty se provádí kontrola přípustnosti konfigurace. Protože konfigurací na této křivce může být nekonečně mnoho (stejně jako bodů), je vhodné tuto křivku diskretizovat. V návrhu algoritmu tedy předpokládám takovou diskretizaci, která převede křivku na konečný počet bodů, ve kterých se bude kontrolovat konfigurace. Body budou spojeny úsečkami o konečné délce a tyto krátké úsečky tedy nahradí kruhový oblouk.

Jak je z pseudokódu patrné, funkce postupně projde hrany a v případě, že se podaří zkonstruovat takovou alternativu, která je kratší nebo rychlejší, nahradí příslušnou část hran touto alternativní cestou. Pokud však není konstrukce alternativního řešení úspěšná, v plánu cesty jsou ponechány obě hrany beze změny.



Obr. 23 Vyhlazení cesty -dvě možné varianty řešení

4.6 Tvorba trajektorie

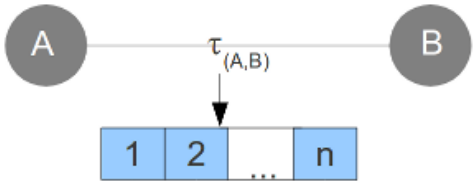
Jak je uvedeno výše v návrhu algoritmu, výstupem mravenčích algoritmů použitých pro hledání cesty je posloupnost hran grafu spojujících start a cílové pozice. Tato posloupnost je dále

zpracována funkcí vyhlazení cesty (viz § 4.5) a z takto upravené trasy je vypočtena trajektorie. Tvorba trajektorie probíhá tak, že na nalezenou a upravenou cestu jsou aplikovány parametry robota, čímž je ke každé pozici na nalezené cestě doplněn další rozměr, a to čas. Během tvorby trajektorie je také možné stanovit rychlostní profil cesty robota daným prostředím.

4.7 Navigace více robotů

Rozšířením plánování cesty robota je případ, kdy se v daném prostředí pohybuje více robotů. Rozšířením výše uvedených algoritmů pro plánování může být řešení problému plánování více robotů v jednom pracovním prostoru. Předpokládejme, že každý z robotů plní samostatný úkol spočívající v přesunu ze startovní pozice do cílových pozic. Plánování cesty *n* robotů v takovéto situaci je možné redukovat na *n* samostatných po sobě jdoucích hledání cesty s následným řešením kolizí.

Dalším přístupem k plánování cesty více robotů může být rozdělení mravenčí kolonie na *n* menších kolonií, přičemž každá provádí hledání cesty pro individuálního robota. Při volbě tohoto přístupu je vhodné zajistit, aby u jednotlivých kolonií nedocházelo k vzájemnému ovlivňování hledání díky feromonovým stopám, které slouží jako komunikační prostředek mravenců. Řešení tohoto problému může spočívat v ukládání více feromonových stop, jak naznačuje obr. 24 . Paralelní hledání cesty více robotů pomocí mravenčích algoritmů také vyžaduje úpravu podmínek ukončení hledání cesty. Podmínky ukončení hledání tak, jak jsou uvedeny v § 4.4.1 a 3.8.1 , předpokládají hledání cesty jen jednoho robota. V případě paralelního hledání je tedy nutné podmínky ukončení vhodně přeformulovat.



Obr. 24 Více feromonových stop na hraně grafu

Po nalezení cesty všech robotů je nutné řešit případné kolize. V návrhu plánování cesty více robotů využiji pro řešení kolizí postup popsaný v kapitole 2.4 , tedy prioritní pravidla. Plánování *n* robotů proběhne následovně:

1. Nejprve bude pro každého robota nalezena cesta, která bude následně na základě parametrů robota převedena na trajektorii.
2. Určím prioritní pravidlo pro všechny roboty a na jeho základě vyberu 1. robota. Pro tohoto robota není nutné řešit kolize, protože má nejvyšší prioritu a trajektorie tohoto robota tedy nebude nijak měněna.
3. Postupně vyberu 2. až n. robota. Porovnám trajektorii tohoto robota s trajektoriemi všech robotů zpracovaných dříve. Pokud bude nalezena kolize, upravím trajektorii robota o nižší prioritě.

Jak je patrné z výše uvedeného, tento přístup nehledá optimální strategii z hlediska doby potřebné pro dokončení úkolů všech robotů, ale řeší pouze případné kolize.

V rámci experimentů s navigací více robotů se zaměřím na porovnání výsledků dvou výše zmíněných přístupů plánován cest robotů. Podmínky ukončení hledání cesty budou definovány následovně: iterace hledání bude znamenat, že bylo nalezeno řešení pro každého z *n* robotů. Počítání nalezených řešení bude společné pro všechny roboty a stagnace bude vyhodnocována jako nejlepší z ukazatelů kvality řešení všech robotů. Při plánování více robotů budu také rozlišovat mezi částečným a úplným řešením, částečné řešení předpokládá nalezení alespoň jedné cesty libovolného robota, tedy počet nalezených řešení bude větší než nula. Úplné řešení pak předpokládá nalezení řešení pro každého robota, tedy čítač cyklů bude mít kladnou nenulovou hodnotu.

4.8 Možné rozšíření algoritmů pro neholonomního robota

Algoritmy výše zmíněné předpokládají pro plánování cesty holonomního robota. Pro plánování cesty neholonomního robota je ovšem nutné algoritmus částečně upravit. Úpravy je možné realizovat v několika směrech. První možností je úprava vytváření grafu popisujícího volný stavový

prostor. Nabízí se možnost vytvářet takový graf, který zohlední parametry robota. Konstrukce řešení v takovémto grafu pak bude probíhat stejným způsobem, jak je uvedeno v algoritmech výše. Tato možnost tedy předpokládá náročnější předzpracování a jednodušší dotazovací část, což by mohlo být výhodné v prostředí se statickými překážkami, kde není nutné předzpracování opakovat.

Druhým možným přístupem je použití jednodušší grafové struktury, např. stejné jako je uvedeno v kapitol 4.2.1 a při plánování cesty kontrolovat přípustnost přechodů mezi konfiguracemi. To může probíhat tak, že bude použita část funkce pro vyhlazování cesty a při konstrukci řešení proběhne dotazování, zda je možné propojit dvě sousední hrany, tedy ve výsledku propojit dva vrcholy – realizovat přechod z jedné konfigurace do druhé, a to tak, aniž by během přesunu došlo k překročení limitů daného robota, či dosažení kolizní konfigurace. Pokud proběhne dotazování úspěšně, je hrana ponechána v grafu a může být použita pro konstrukci řešení, pokud ale dotazování na proveditelnost není úspěšné, nabízí se možnost hranu z grafu odstranit. Parametr R použitý ve vyhlazování cesty, zde pak v dotazovací části, může být v případě neholonomního robota úspěšně nahrazen minimálním poloměrem zatáčení.

Další možný přístup k řešení problému spočívá v použití algoritmu pro plánování cesty holonomního robota s rozšířením v podobě uchovávání všech nalezených řešení. Po ukončení hledání by byly všechna nelezení řešení seřazena podle kvality a postupně by probíhalo dotazování, zda může být konkrétní řešení použito pro navigaci neholonomního robota. Po nalezení prvního vyhovujícího by došlo k ukončení dotazování. Pokud by řešení nalezeno nebylo, nabízí se možnost znovu spustit hledání cesty.

Algoritmy navrhované v této podkapitole nejsou v rámci této práce realizovány. Jedná se o nástin toho, jakým směrem by bylo možné rozšířit tuto práci v budoucnu.

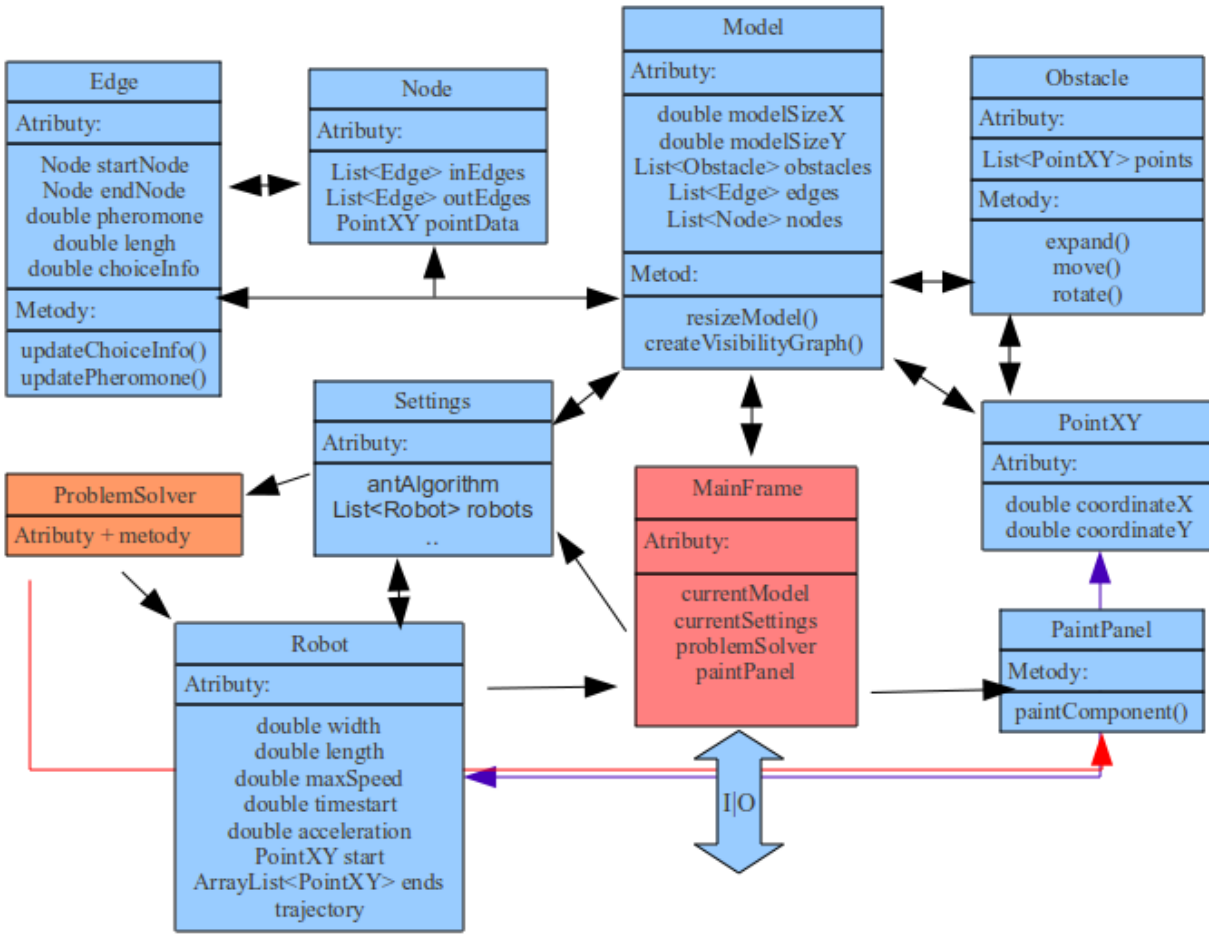
5 IMPLEMENTACE ALGORITMŮ A POPIS APLIKACE

Simulační prostředí robota je vytvořeno v programovacím jazyce Java. Simulační prostředí umožňuje vytváření a editaci prostředí robota, dále simulaci robota, či robotů v daném prostředí a zpracování a export výsledků plánování cesty.

5.1 Členění aplikace

Program je vytvořen z desíti hlavních tříd, jak je naznačeno na obr. 25 . Hlavní třídou programu je třída *MainFrame*, která zajišťuje zpracování vstupů a částečně i výstupů. Metody této třídy dále spouští vlastní hledání cesty.

Pracovní prostředí robota představuje třída Model. Tato třída obsahuje pole překážek třídy *Obstacle* a z těchto překážek vytváří po jejich expanzi graf viditelnosti, daný vrcholy třídy *Node* a hranami třídy *Edge*. Při vytváření grafu viditelnosti pro řešení daného problému jsou použity cílové a startovní vrcholy všech robotů, pro které bude hledaná cesta. Jak je z obr. 25 patrné, atributy hran grafu jsou kromě startovních a cílových vrcholů i proměnné určující množství umělého feromonu na hranách a dále proměnná *choiceinfo*. Tato proměnná je použita pro urychlení výpočtů při hledání cesty, a její hodnota je dána jako $choiceinfo_{(i,j)}=[\tau_{(i,j)}]^{\alpha}[\eta_{(i,j)}]^{\beta}$, τ,α,β,η viz § 4.3 . Zavedení této proměnné vychází z faktu, že všichni umělí mravenci potřebují při každém kroku tvorby řešení hodnotu součinu heuristické informace a intenzity feromonové stopy. Z důvodu snížení výpočetní náročnosti se tedy zavádí tato proměnná, přičemž hodnota této proměnné se aktualizuje na všech hranách po každém provedeném cyklu hledání – aktualizaci provádí instance tříd *Updater* a *lowerThread* uvedené níže.



Obr. 25 Zjednodušený diagram tříd

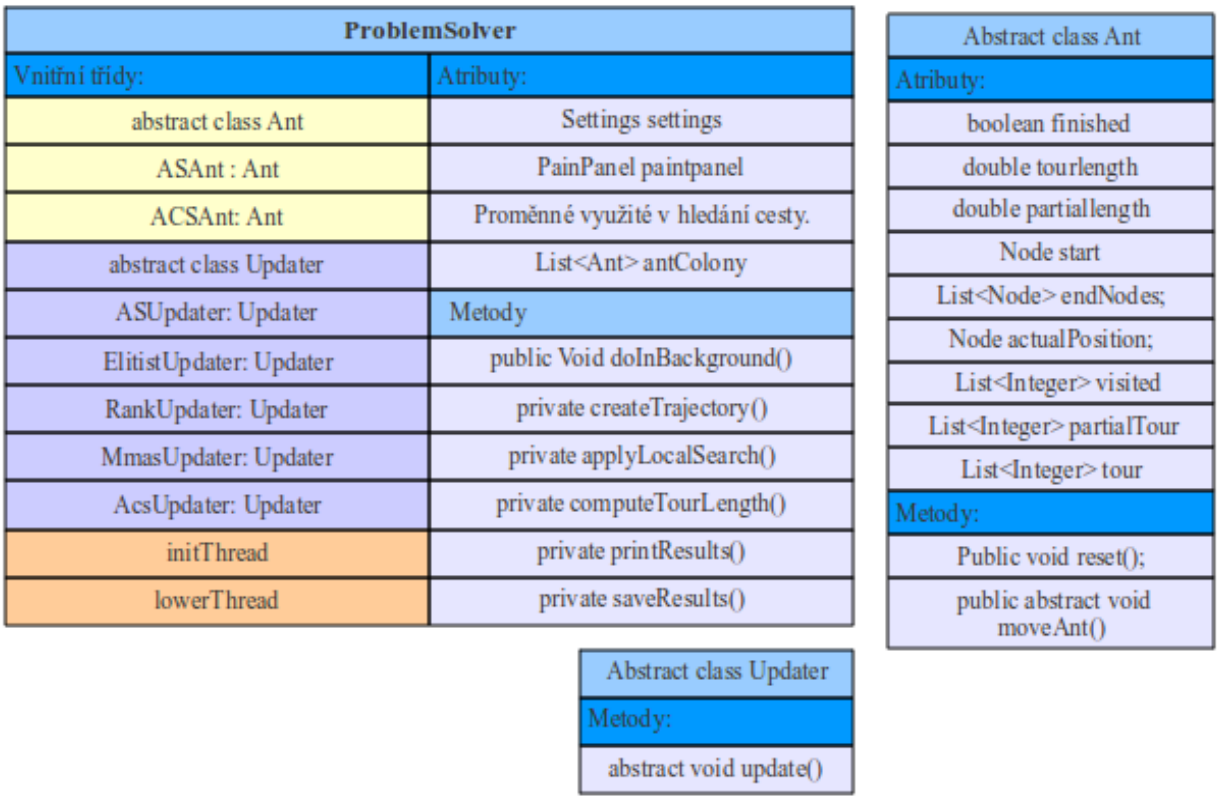
Překážky, startovní a cílové body robotů jsou definovány body třídy *PointXY*. Tato třída je také použita pro určení pozice vrcholu grafu v pracovním prostoru.

Zobrazování výsledků hledání a vizualizaci pracovního prostoru zajišťuje třída *PaintPanel*. Instance této třídy nijak nekomunikuje s ostatními komponenty programu a jen vykresluje data předané ostatními komponenty.

Vlastní hledání cesty je realizováno instancí třídy *ProblemSolver* vytvořené ve třídě *MainFrame*. Parametry hledání, tedy počet mravenců použitých pro řešení problému, zvolený algoritmus hledání, parametry α, β, ρ, aj. (viz § 4.4), počet robotů a způsob hledání v případě více robotů, podmínky ukončení hledání, počet opakování, aj. předává instance třídy *Settings*.

Instance třídy *Robot* představují jednotlivé roboty, pro které jsou plánovány cesty daným prostředím. Parametry robotů, zde atributy třídy *Robot,* jsou přes instanci třídy *Setting* předány do výkonného jádra programu zajišťujícího vlastní plánování cesty robota a tvorbu trajektorií.

Na obr. 26 je naznačeno zjednodušené schéma třídy *ProblemSolver*. Tato třída představuje výkonné jádro programu, přičemž veškeré hledání cesty a tvorby trajektorie robota se odehrává právě v instanci této třídy.



Obr. 26 Třída ProblemSolver

V rámci této třídy je definováno několik vnitřních tříd, tyto třídy lze rozdělit do tří skupin. První skupina představuje abstraktní třídu *Ant* a z ní odvozené *ASAnt* a *ACSAnt.* Instance těchto tříd – kolonie umělých mravenců, konstruují řešení daného problému. Rozdíl mezi třídou *ASAnt* a *ACSAnt* spočívá v použitém algoritmu pro řešení daného problému. Zatímco instance třídy *ASAnt* používají pro řešení problému plánování cesty varianty algoritmu *Ant system* (viz kapitoly 3.5.1 – 3.5.4), instance třídy ACSAnt používají pro daného problému algoritmus *Ant Colony System* (viz § 3.5.5).

Další skupinou vnitřních tříd ve třídě *problemSolver* jsou třídy odvozené od abstraktní třídy *Updater*. Tato třída zajišťuje během hledání cesty aktualizace feromonových stop a proměnné *choceinfo* na hranách grafu. Každá z odvozených tříd zde zastupuje jeden z algoritmů použitých pro plánování cesty.

Poslední skupinou vnitřních tříd definovaných ve třídě *ProblemSolver* jsou třídy *initThread* a *lowerThread.* Instance těchto tříd provádí ve více vláknech paralelně odpařování, resp. inicializaci feromonových stop, instance třídy *lowerThread* dále provádí aktualizaci proměnné *choceinfo*. Instance třídy *lowerThead* jsou volány v každém cyklu algoritmu pro hledání cesty. Instance třídy *initThread*

jsou volány vždy, když je třeba inicializovat feromonové stopy na grafové struktuře, tedy při každém opakování hledání cesty.

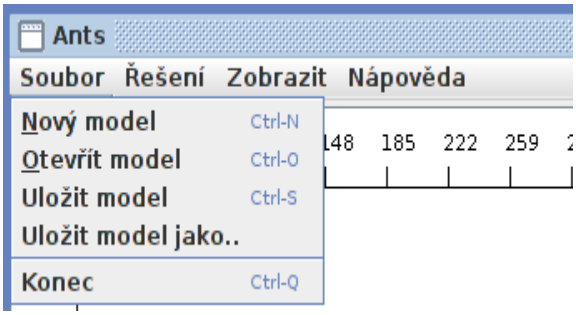
Plánování cesty robota probíhá v této aplikaci následujícím způsobem:

- Vznikne instance třídy *Model*, v rámci této instance je vytvořena kolekce překážek třídy *Obstacle.*
- Vznikne instance třídy *Settings,* kde jsou specifikovány parametry hledání cesty. Výše zmíněná instance třídy *Model* je přidána do instance třídy *Setting.* Dále vznikne jeden, či více robotů – instance třídy *Robot*. Tito roboti mají zadané startovní a cílové body a nastaveny parametry definující jednotlivé roboty. Trajektorie jsou prázdné. Všichni roboti jsou vloženi do výše zmíněné instance třídy *Settings.*
- Instance třídy *Settings* sdružující všechny parametry hledání cesty, model prostředí a roboty, pro které se bude hledat cesta, je předána do nově vytvořené instance třídy *ProblemSolver.* Vlastní hledání cesty je spuštěno ze třídy *MainFrame.* Hledání cesty je zahájeno voláním funkce *doInBackground()*. *V*oláním této funkce se spustí proces na pozadí aplikace a v instanci třídy *ProblemSolver* je vytvořen z modelu prostředí graf viditelnosti, jsou do něj přidány startovní a cílové body robotů, na grafu je dále provedena inicializace feromonových stop. Dále je vytvořena kolonie umělých mravenců příslušné třídy a je vytvořena instance třídy *Updater*, opět v závislosti na zvoleném algoritmu hledání.
- Je spuštěno vlastní hledání cesty, po dokončení hledání jsou výsledky předány k dalšímu zpracování. Je vytvořena trajektorie, vyřešeny případné kolize (v případě plánování cesty více robotů) a výsledky jsou zobrazeny a v případě požadavku uloženy do souboru. Pokud je zadáno opakování hledání cesty, je reinicializovaná grafová struktura, je vytvořena nová kolonie umělých mravenců a instance třídy *Updater*, jsou vymazány dosavadní výsledky a je znovu spuštěno hledání cesty.

5.2 Popis aplikace

Vytvořená aplikace umožňuje tvorbu modelu prostředí a plánování cesty robota tímto prostředím. Snahou při návrhu této aplikace bylo dosáhnout snadného a intuitivního ovládání.

Hlavní menu aplikace umožňuje vytváření nového modelu pracovního prostředí robota, dále pak export modelu do souboru a otevření modelu ze souboru. Hlavní menu dále obsahuje volby pro zobrazení panelu s parametry řešení a možnost nahrát ze souboru uložené řešení. V případě, že již proběhlo hledání cesty, zviditelní se v hlavním menu volby pro úpravu zobrazení zkonstruovaného řešení. Poslední volbou dostupnou z hlavního menu je nápověda.

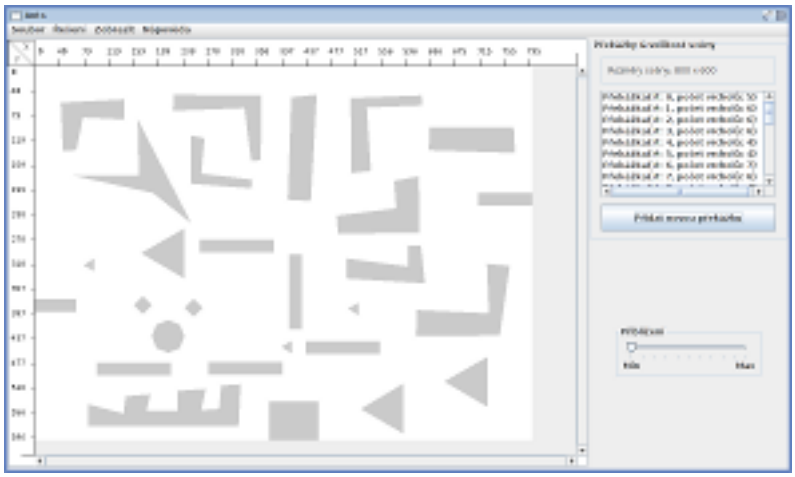


Obr. 27 Hlavní menu programu

Okno aplikace (viz obr. 28) je rozděleno na dvě části, větší slouží jako pracovní plocha robota, menší pak slouží k zobrazování panelů s nastavením. Levý panel má pevně dánu šířku, a to 300 bodů. Velikost pracovní plochy, stejně tak okna programu, není pevně ohraničena, po každém spuštění programu je velikost okna nastavena podle velikosti monitoru. Velikost pracovní plochy lze měnit, a to libovolně v rozmezí od 400 x 400 bodů do 3000 x 3000 bodů. Levý panel také nabízí možnost přiblížení pracovní plochy a to tažením příslušným posuvníkem. Pracovní plocha je pro větší přehlednost po obvodu orámovaná pravítky.

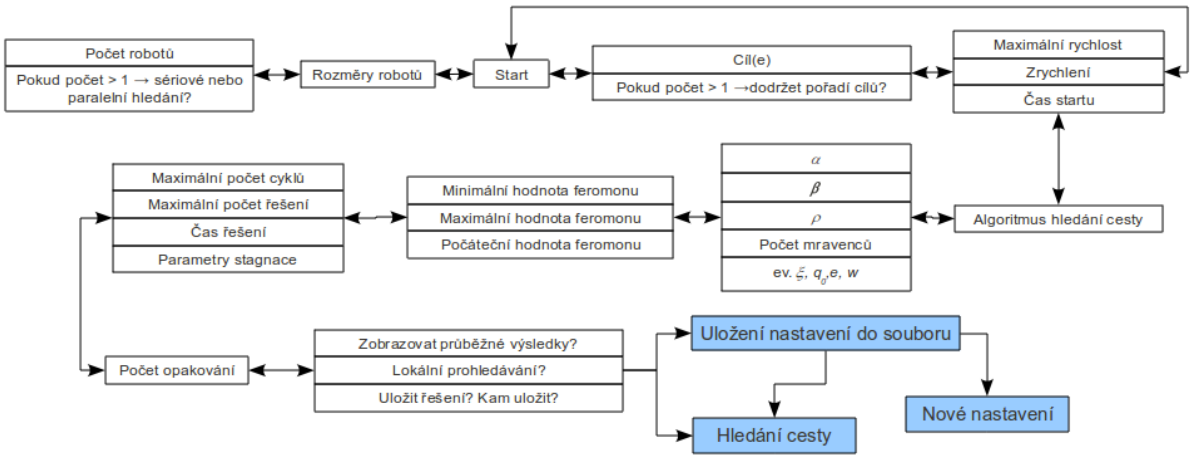
Po spuštění programu má uživatel možnost otevřít uložený model prostředí, nebo vytvořit novou scénu. Vkládání překážek do pracovní plochy je možné dvojím způsobem, jednak lze vytvořit

vlastní tvary polygonálních překážek, přičemž jsou povoleny jak konvexní, tak konkávní překážky. Jediné omezení nepřípustné překážky jsou sebeprotínající. Druhou možností vkládání překážek je výběr z některého z předdefinovaných tvarů. K možnosti přidat překážku se uživatel dostane po stisku příslušného tlačítka, přičemž je zobrazeno menu s volnou předdefinované překážky nebo ručního zadání. Překážky lze v pracovním prostředí robota posouvat, rotovat a případně mazat a to přes panely a menu přístupná ze seznamu překážek na úvodním panelu.



Obr. 28 Okno aplikace , pracovní prostor robota s překážkami

Nastavení parametrů hledání cesty robota je možné získat dvojím způsobem. Jednak lze nastavení hledání cesty nahrát ze souboru, a to i s příslušným modelem prostředí, ve kterém se má hledání odehrávat. Druhým krokem je vytvoření nového nastavení hledání. Parametry hledání lze zadat na příslušných panelech, které jsou přístupné pře hlavní menu programu *(Řešení → Nastavit parametry řešení)*. Na panelu v levé části okna aplikace jsou postupně zobrazeny příslušná pole na zadání parametrů řešení, a to v pořadí, jaké naznačuje obr. 29 . V případě, že se uživatel kdykoli během zadávání parametrů hledání rozhodne přerušit zadávání těchto parametrů, má možnost příslušný panel zavřít a opět je zobrazen panel se seznamem překážek a uživatel má možnost úpravy pracovního prostředí robota. Pokud se poté rozhodne pokračovat v zadávání parametrů řešení, je mu nabídnuto pokračování na místě, kde skončil.

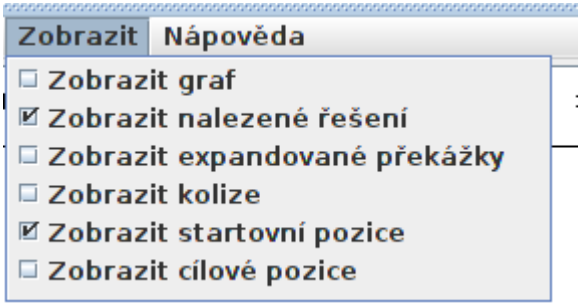


Obr. 29 Schéma zadávání parametrů řešení

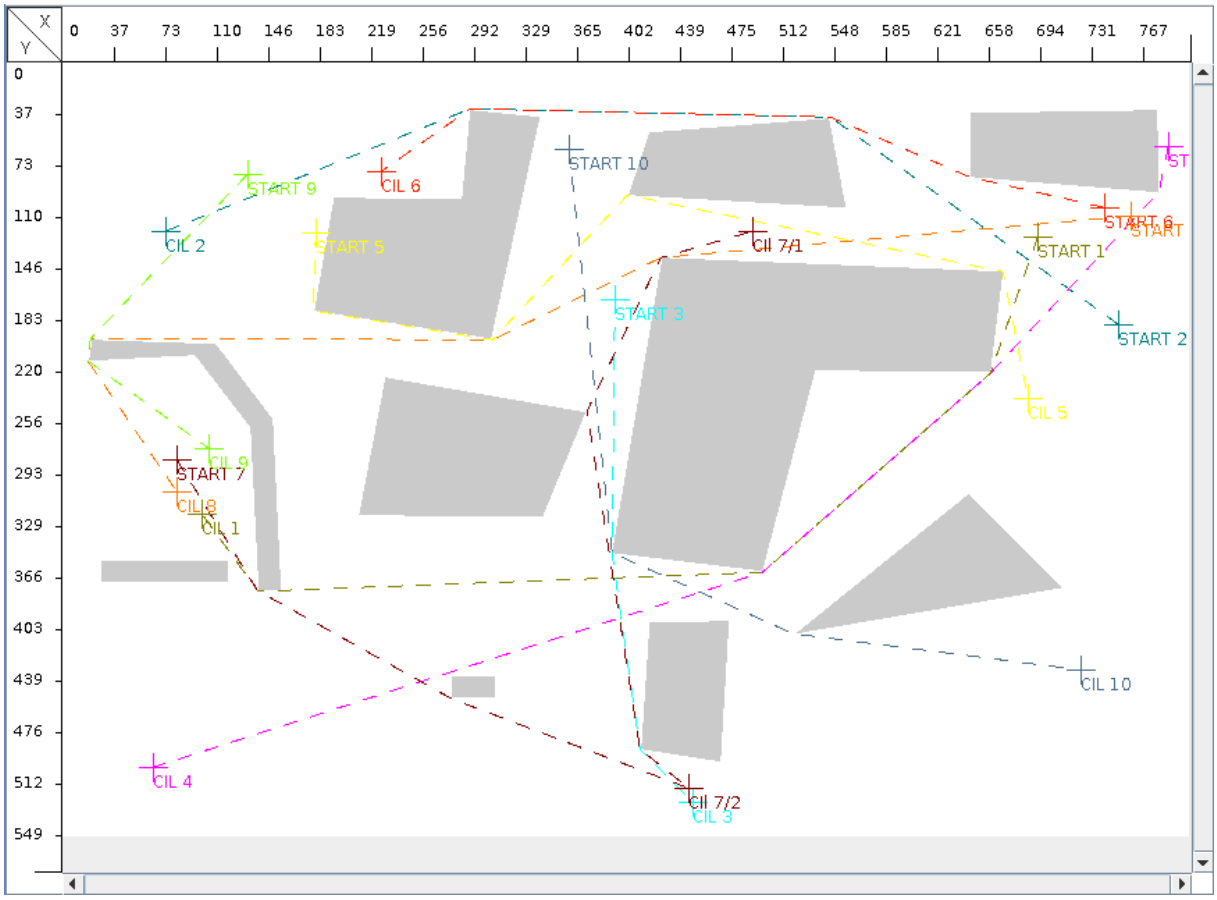
V okamžiku, kdy jsou zadány všechny parametry hledání cesty nebo je načten soubor s parametry hledání, je na panelu v levé části aplikace zviditelněno tlačítko, kterým lze spustit hledání cesty. Jak je uvedeno výše v § 5.1 , hledání je spuštěno v samostatném výpočetním vlákně běžícím na pozadí aplikace. Hledání cesty lze kdykoli přerušit stiskem stejného tlačítka, které bylo použito pro zahájení hledání.

Po ukončení hledání jsou na pracovní plochu robota vykresleny cesty všech robotů, jsou zvýrazněné body, kde byly řešeny kolize a v hlavním menu je zpřístupněno rozšiřující nastavení –

viz obr. 30 . Pokud byla při zadávání parametrů řešení zadána složka pro uložení výsledků, je v této složce vytvořen soubor, do kterého jsou exportovány výsledky hledání cesty a je do něj vložen snímek pracovního prostoru robota zachycující výsledky hledání. Po ukončení hledání je možnost pokračovat v tvorbě modelu prostředí a případném dalším hledání cesty.



Obr. 30 Rozšířené hlavní menu po ukončení řešení



Obr. 31 Pracovní plocha robota s nalezenými cestami

6 VÝSLEDKY EXPERIMENTŮ

Experimenty probíhaly ve více krocích. Nejprve byla ověřována funkčnost jednotlivých komponent programu a funkčnost programu jako celku. V dalším kroku bylo zkoumáno chování algoritmů pro různé kombinace parametrů nastavení, přičemž tyto experimenty byly provedeny bez funkce lokálního prohledávání. Cílem těchto experimentů bylo nalezení takových parametrů, pro které poskytuje daný algoritmus nejlepší výsledky. Parametry nastavení, pro které bylo dosaženo nejlepších výsledků, byly použity v následujících etapách experimentů. Dalším krokem byla aplikace procedury lokálního prohledávání, přičemž experimenty proběhly jak s funkcí lokálního prohledávání, tak i bez této funkce, výsledky obou těchto testů byly následně porovnány. Poslední etapou experimentů bylo plánování cesty více robotů. Zde byly porovnávány výsledky plánování cesty při sériovém a paralelním hledání.

V rámci ověřování funkčnosti programu bylo zjištěno, že funkce navrhovaná v podkapitole 4.5 nefunguje správně. Důvod, proč tomu tak je, se nepodařilo odhalit, a pro zajištění správného fungování ostatních částí programu byla tato funkce odebrána, a není tedy v programu implementována. Vzhledem k tomu, že část této funkce využívala i procedura tvorby trajektorie a časového plánu pohybu robota, není tedy v programu ani tato problematika řešena.

6.1 Parametry nastavení a porovnání algoritmů

Prvním krokem experimentů bylo stanovení takových parametrů, pro které dává daný algoritmus nejlepší výsledky, a dále vzájemné porovnání algoritmů. Vzhledem k tomu, že parametry hledání lze na nastavovat v mnoha kombinacích, byly jako výchozí použity parametry, které uvádí literatura [10]. Parametry byly testovány na jednoduchém modelu prostředí o rozměrech 400 x 400 bodů se 16 konvexními překážkami. Volný pracovní prostor tohoto modelu je popsán grafem o 54 vrcholech a 456 hranách, viz obr. 32 . Startovní bod byl umístěn na souřadnice (25 ; 25) a dále byly do modelu vloženy 3 cílové body na souřadnicích (180 ; 380), (200 ; 25) a (190 ; 210). Tři cílové body byly vloženy pro ověření funkčnosti plánování cesty do více cílových bodů.

Všechny experimenty realizované na tomto modelu prostředí měly nastaveny ukončení hledání cesty po dosažení alespoň jedné z následujících podmínek:

- Dosažení 3000 nalezených řešení.
- Dosažení 1000 iterací.
- Časový limit řešení 10 sekund.
- Ustálení řešení v rozmezí 5 % během 500 iterací.

Sledovanými parametry řešení byly:

- Délka nejlepšího nalezeného řešení.
- Počet nalezených řešení.
- Vývoj řešení v čase, stagnace – byla sledovaná pomocí podmínky ukončení běhu programu.
- Podíl úspěšných řešení, tedy takových, kdy došlo k nalezení cíle.

Jak je uvedeno výše, parametry mravenčích algoritmů je možné nastavit v mnoha kombinacích. Na základě údajů v literatuře [10] a průběžných výsledků experimentů byly zvoleny parametry pro jednotlivé mravenčí algoritmy (viz tab 1. –tab 9.), které byly dále ověřovány na uvedeném modelu. Pro každou kombinaci parametrů bylo realizováno celkem 100 experimentů, z výsledků všech experimentů byl posléze stanoven aritmetický průměr. Dílčí výsledky jednotlivých experimentů dále umožňují stanovit časový vývoj řešení.

	1	2	3	4	5	6	7
α	1	2	1	1	1	1	1
β	3	3	2	2	2	2	2
ρ	0,5	0,5	0,5	0,7	0,9	0,9	0,5
N	15					30	45

Tab 1. Nastavení parametrů řešení pro Ant system

Parametry α, β v tab 1. určují důležitost feromonových stop, respektive heuristické informace, ρ je koeficient odpařování feromonových stop. Parametr N určuje velikost mravenčí

kolonie použité pro konstrukci řešení.

Experimenty provedené podle kombinací nastavení z tab 1. dosáhly následujících výsledků:

Kombinace parametrů	1	2	3	4	5	6	7
Doba řešení [ms]	236	271	231	224	236	220	153
Iterace	206	208	224	221	243	137	75
Nalezené cesty	3006	3008	3007	3007	3006	3013	3021
Stagnace [%]	0	0	1	0	0	0	0
Úspěšné hledání [%]	100	100	100	100	100	100	100
Nejlepší řešení	696	676	667	668	660	659	659
Průměrná délka všech řešení	814	791	898	871	960	1091	890

Tab 2. Výsledky řešení pomocí algoritmu Ant system

Z výše uvedené tabulky je patrné, že algoritmus nalezl řešení ve všech případech, a to bez toho, aniž by vykazoval stagnující chování. Nejlepších výsledků daný algoritmus dosáhl pro kombinace 6) a 7), tj. pro hodnoty $\alpha = 1$, $\beta = 2$, $\rho = 0{,}5$ resp. 0,9. Výsledky řešení pomocí algoritmu Ant system byly použity v nastavení parametrů dalších algoritmů.

	1	2	3	4	5	6	7	8
α	1	1	1	1	1	1	1	1
β	2	2	2	2	2	2	2	2
ρ	0,5	0,9	0,9	0,5	0,5	0,9	0,5	0,9
e	2	2	4	4	6	6	10	10

Tab 3. Nastavení parametrů řešení pro Elitist ant system

Hodnoty parametrů α, β a ρ byly převzaty z nejúspěšnějších řešení pomocí algoritmu Ant system. Velikost kolonie řešící problém plánování cesty byla nastavena na 15 agentů. Parametr *e* udává dodatečné posílení feromonové stopy na nejlepším řešení (viz § 4.4.3).

Hledání cesty pomocí algoritmu Elitist ant system na základě parametrů z tab 3. dosáhlo následujících výsledků:

	1	2	3	4	5	6	7	8
Doba řešení [ms]	224	206	232	211	218	208	195	156
Iterace	217	217	228	214	212	226	210	79
Nalezené cesty	3006	3007	3002	3007	3006	3007	3007	3022
Stagnace [%]	0	0	0	0	0	0	0	0
Úspěšné hledání [%]	100	100	100	100	100	100	100	100
Nejlepší řešení	657	659	661	656	657	660	657	659
Průměrná délka všech řešení	804	808	803	773	756	791	737	870

Tab 4. Výsledky řešení pomocí algoritmu Elitist ant system

Jak je z tab 4. a tab 2. patrné, Elitist ant system dosahuje při řešení daného problému přibližně stejných výsledků jako algoritmus Ant system, nepatrný rozdíl ve výsledcích těchto algoritmů lze spatřovat v průměrné délce všech řešení, kde Elitist ant system dosáhl nižších hodnot. Důvodem může být posilování feromonových stop na nejkratším nalezeném řešení, tedy postupném převažování jednoho řešení a omezování prozkoumávání pracovního prostoru.

Dalším algoritmem použitým pro konstrukci řešení je Rank-based ant system. Parametry nastavení tohoto algoritmu byly stanoveny následovně. Všechny experimenty s algoritmem Rank-based ant system byly provedeny s kolonií umělých mravenců o 15 členech.

	1	2	3	4	5	6
α	1	1	1	1	1	1
β	2	2	2	2	2	2
ρ	0,9	0,9	0,9	0,95	0,95	0,95
w	2	5	10	2	5	10

Tab 5. Nastavení parametrů pro Rank-based ant system

Parametr *w* je dán jako maximální počet mravenců, kteří mohou v daném kroku řešení uložit umělou feromonovou stopu, viz § 3.5.3 . Výsledky plánování cesty robotu pomocí algoritmu Rank-based ant system jsou uvedeny v tab 6.

Kombinace parametrů	1	2	3	4	5	6
Doba řešení [ms]	220	203	605	225	655	582
Iterace	234	223	584	265	616	578
Nalezené cesty	3006	3007	2454	3007	2064	2477
Stagnace [%]	0	0	89	0	95	81
Úspěšné hledání [%]	100	100	100	100	100	100
Nejlepší řešení	658	661	657	658	658	657
Průměrná délka všech řešení	800	769	1695	862	1753	1702

Tab 6. Výsledky plánování cesty pomocí algoritmu Rank-based ant system

Jak je z výše uvedené tabulky patrné, z hlediska délky řešení dosáhl algoritmus Rank-based ant system přibližně stejných výsledků jako dva výše uvedené algoritmy. Rozdíl spočívá především v ukazatelích stagnace a průměrné délce řešení, pravděpodobná příčina tohoto rozdílu je nevhodně zvolená hodnota parametru *w*.

Dalším algoritmem použitým v experimentech byl Max-min ant system. Hlavními parametry ovlivňující výsledky tohoto algoritmu jsou horní a dolní přípustná hodnota feromonové stopy. Určení těchto hodnot probíhalo experimentálně. Výchozí hodnotou byla intenzita τ_{MAX} určená jako $\tau_{MAX}=\frac{1}{(\rho \cdot L^{nn})}$, kde L^{nn} je nejmenší vzájemná vzdálenost startovních a cílových bodů, jak uvádí [10]. Hodnota τ_{MIN} byla následně určena jako podíl hodnoty τ_{MAX}. Experimenty probíhaly ve více krocích, kdy byla nejprve určena hodnota horní a dolní přípustné intenzity feromonové stopy, pro tyto hodnoty byly provedeny experimenty, a na základě výsledků experimentů byla experimentálně stanovena nová hodnota. Cílem tohoto postupu bylo nalézt takovou hodnotu uvedených parametrů, která zajistí řešení o kvalitě obdobné jako v tab 2. –tab 6. . Tento postup byl ukončen po sedmi krocích, protože algoritmus začal vykazovat opětovné zhoršení výsledků.

	1	2	3	4	5	6	7
α	1	1	1	1	1	1	1
β	2	2	2	2	2	2	2
ρ	0,98	0,98	0,98	0,98	0,98	0,98	0,98
τmax	0,015	0,015	0,015	0,015	0,025	0,05	0,06
τmin	0,005	0,01	0,0075	0,012	0,015	0,035	0,04

Tab 7. Parametry nastavení algoritmu Max-min ant system

Kombinace parametrů	1	2	3	4	5	6	7
Doba řešení [ms]	661	618	632	995	1171	978	952
Iterace	643	621	630	600	549	599	593
Nalezené cesty	1192	1148	1162	2067	2761	2347	2255
Stagnace [%]	96	96	99	96	64	88	88
Úspěšné hledání [%]	100	100	100	100	100	100	100
Nejlepší řešení	673	673	672	668	665	667	668
Průměrná délka všech řešení	2004	2003	1999	1979	1972	1970	1975

Tab 8. Výsledky řešení pomocí algoritmu Max-min ant system

Jak je z výše uvedené tabulky patrné, algoritmus Max-min ant system dosáhl horších výsledků než výše uvedené algoritmy. Důvodem mohou být nevhodně zvolené hodnoty τ_{MAX} a τ_{MIN}, případně nízký časový limit řešení.

Posledním ze skupiny implementovaných a porovaných algoritmů je Ant colony system. Parametry tohoto algoritmu jsou dány následující tabulkou:

	1	2	3	4	5	6	7
β	2	2	2	3	3	2	2
ρ	0,98	0,98	0,98	0,9	0,9	0,9	0,95
q0	0,1	0,1	0,1	0,1	0,9	0,9	0,9
ξ	0,9	0,5	0,1	0,9	0,1	0,1	0,1

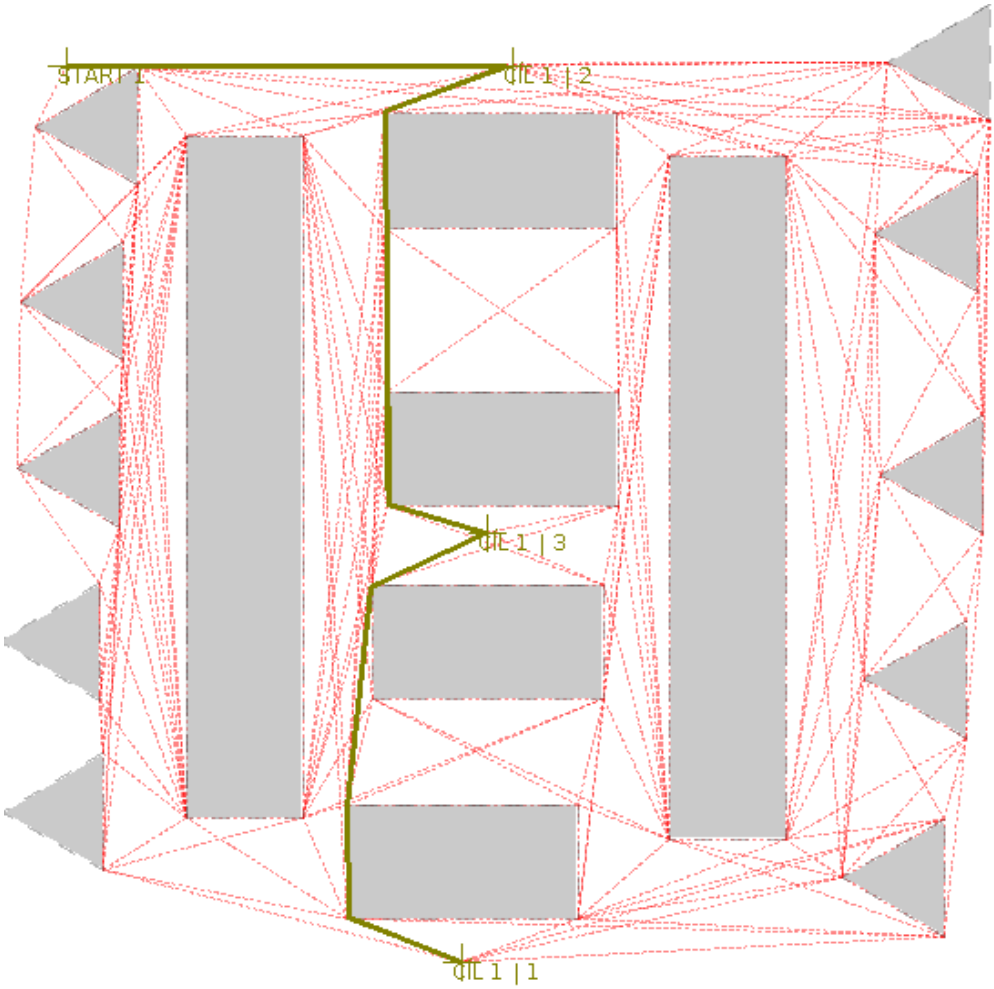
Tab 9. Parametry nastavení algoritmu Ant colony system

Hodnoty parametrů v tab 9. byly stanoveny na základě informací v literatuře [10], dále pak na základě průběžných výsledků experimentů. Výsledky experimentů s algoritmem Ant colony system jsou uvedeny v tabulce níže.

Kombinace parametrů	1	2	3	4	5	6	7
Doba řešení [ms]	2431	509	384	2739	415	877	921
Iterace	703	647	632	716	617	524	527
Nalezené cesty	1272	1869	1267	1173	1251	2897	2892
Stagnace [%]	92	94	96	86	97	49	51
Úspěšné hledání [%]	100	100	100	100	100	100	100
Nejlepší řešení	767	683	675	775	667	664	662
Průměrná délka všech řešení	3196	1933	1989	3191	1989	1941	1932

Tab 10. Výsledky řešení pomocí algoritmu Ant colony system

Jak je z tab 10. patrné, výsledky algoritmu Ant colony system vykazují přibližně stejné trendy jako výsledky algoritmu Max-min ant system, tedy velkou průměrnou délku řešení, vysokou míru stagnace a dlouho dobu řešení daného problému. Důvodem proč tomu tak je, může být jedna ze základních vlastností těchto algoritmů, a to vysoká míra prohledávání stavového prostoru.



Obr. 32 Model prostředí s nalezenou cestou

Porovnáním výsledků v tab 2. –tab 10. je patrné, že původní předpoklad, tedy že Max-min ant system a Ant colony system budou dosahovat nejlepších výsledků, se nenaplnil. Nejlepších výsledků mezi uvedenými algoritmy dosahuje Elitist ant system. Přestože se původní předpoklady nepotvrdily, všechny výše uvedné algoritmy prokázaly schopnost řešit daný problém. Důležitým zjištěním také je, že při všech experimentech bylo nalezeno alespoň jedno řešení.

6.2 Lokální prohledávání

Další etapou experimentů s danými algoritmy byla aplikace funkce lokálního prohledávání, a porovnání výsledků řešení s a bez aplikace této funkce. Cílem bylo stanovit, za jakých podmínek je výhodné použít funkci lokálního prohledávání, a kdy naopak její použití není nezbytně nutné. Funkce byla použita v kombinaci se všemi výše uvedenými algoritmy. Parametry nastavení daných algoritmů jsou následující:

- Ant system: $\alpha = 1$; $\beta = 2$; $\rho = 0{,}5$.
- Elitist ant system: $\alpha = 1$; $\beta = 2$; $\rho = 0{,}5$; $e = 4$.
- Rank-based ant system: $\alpha = 1$; $\beta = 2$; $\rho = 0{,}9$; $w = 10$.
- Max-min ant system: $\alpha = 1$; $\beta = 2$; $\rho = 0{,}98$; $\tau_{MAX} = 0{,}06$ a $\tau_{MIN} = 0{,}04$.
- Ant colony system: $\beta = 2$; $\rho = 0{,}95$; $q0 = 0{,}1$; $\xi = 0{,}1$.

Jedná se tedy o hodnoty, pro které daný algoritmus dosáhl v první etapě experimentů nejlepších výsledků. Tyto hodnoty byly použity pro řešení problému plánování cesty v modelu prostředí o rozměrech 1000 x 1000 bodů s 57 konvexními a konkávními překážkami, viz obr. 33 . V tomto modelu prostředí probíhalo plánování cesty mezi startem na pozici (80 ; 725) a cílem o souřadnicích (910 ; 95). Všechny uvedené algoritmy byly postupně použity pro konstrukci cesty za následujících podmínek ukončení hledání řešení:

1. Doba řešení 20 sekund, maximální počet iterací 20000, maximální počet cest 20000.
2. Doba řešení 10 sekund, maximální počet iterací 10000, maximální počet cest 10000.
3. Doba řešení 5 sekund, maximální počet iterací 5000, maximální počet cest 5000.
4. Doba řešení 3 sekundy, maximální počet iteraci 3000, maximální počet cest 3000.
5. Doba řešení 1 sekunda, maximální počet iterací 1000, maximální počet cest 1000.

Výsledky hledání cesty pro obě kombinace aplikace funkce lokálního prohledávání jsou dány následujícími tabulkami.

	Bez aplikace lokálního prohledávání					S aplikací lokálního prohledávání				
Konec řešení [ms]	1000	3000	5000	10000	20000	1000	3000	5000	10000	20000
Doba řešení [ms]	1001	1894	2455	4451	7080	1001	2170	3298	6461	12272
Iterace	89	204	334	667	1124	78	210	334	671	1265
Nalezené cesty	1134	3005	5010	10005	17826	1168	3000	5010	10004	18802
Stagnace [%]	0	0	0	0	6	0	0	0	0	4
Úspěšné hledání [%]	100	100	100	100	100	100	100	100	100	100
Nejlepší řešení	1465	1419	1434	1448	1453	1407	1404	1400	1371	1373
Průměrná délka všech řešení	4341	3733	3244	2871	2679	3035	2740	2597	2531	2487

Tab 11. Ant system a aplikace lokálního prohledávání

	Bez aplikace lokálního prohledávání					S aplikací lokálního prohledávání				
Konec řešení [ms]	1000	3000	5000	10000	20000	1000	3000	5000	10000	20000
Doba řešení [ms]	547	1219	2024	3791	7007	820	1940	3429	5524	13711
Iterace	67	206	334	672	1201	85	233	341	673	1276
Nalezené cesty	1005	3000	5010	10005	18065	1005	3000	5010	10005	19145
Stagnace [%]	0	0	0	0	6	0	0	0	0	4
Úspěšné hledání [%]	100	1000	100	100	100	100	100	100	100	100
Nejlepší řešení	1452	1431	1425	1388	1352	1352	1345	1334	1325	1327
Průměrná délka všech řešení	3537	2738	2540	2328	2284	2816	2393	2287	2293	2194

Tab 12. Elitist ant system a aplikace lokálního prohledávání

	Bez aplikace lokálního prohledávání					S aplikací lokálního prohledávání				
Konec řešení [ms]	1000	3000	5000	10000	20000	1000	3000	5000	10000	20000
Doba řešení [ms]	1001	3005	5004	4266	8342	820	1940	4323	5524	13711
Iterace	68	208	349	706	1261	89	245	358	707	1340
Nalezené cesty	819	2287	3831	10505	18968	1055	3150	5008	10006	18453
Stagnace [%]	0	0	0	0	6	0	0	0	0	5
Úspěšné hledání [%]	100	100	100	100	100	100	100	100	100	100
Nejlepší řešení	1457	1447	1438	1446	1412	1391	1381	1369	1356	1347
Průměrná délka všech řešení	3572	2765	2565	2351	2307	2844	2417	2310	2316	2216

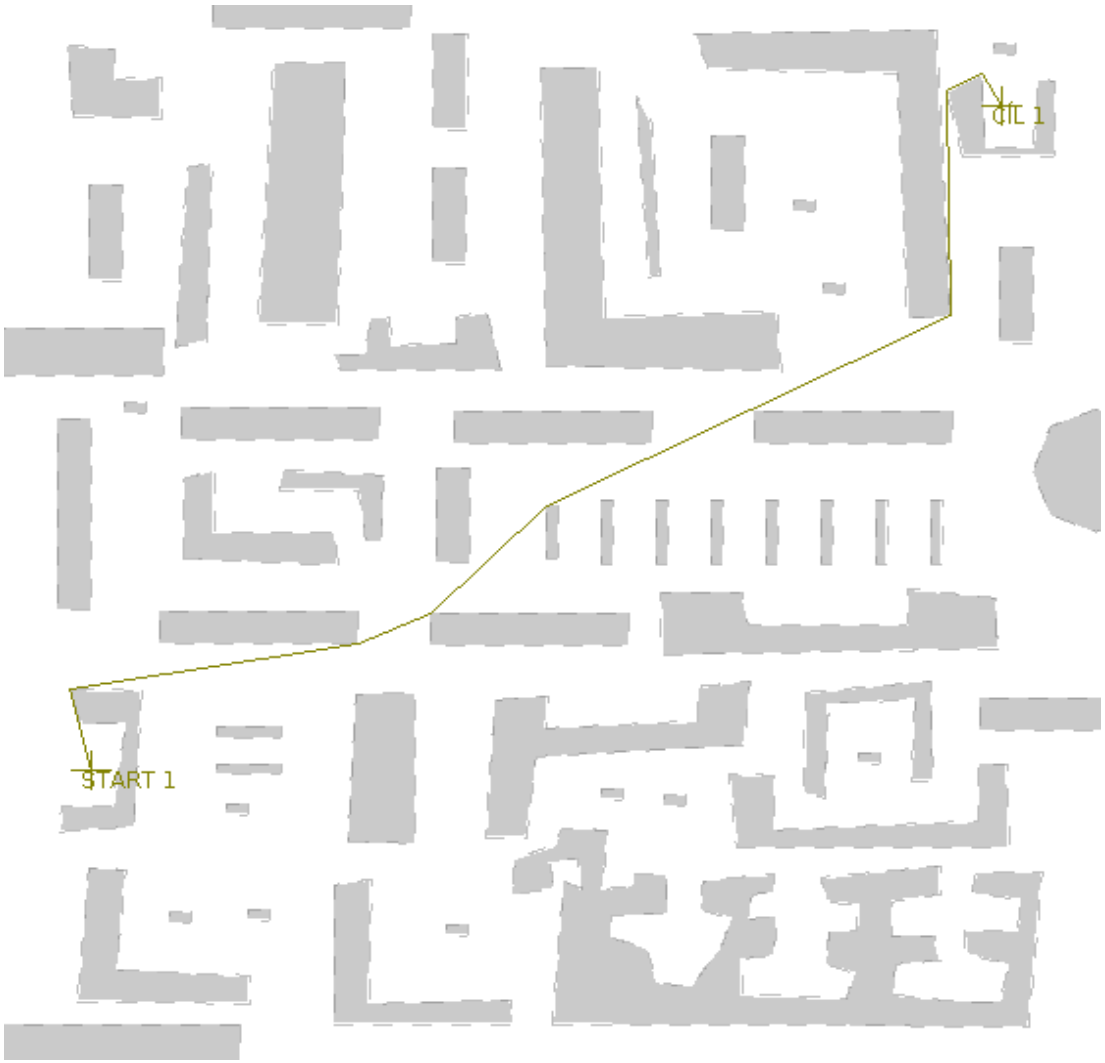
Tab 13. Rank-based ant system a aplikace lokálního prohledávání

	Bez aplikace lokálního prohledávání					S aplikací lokálního prohledávání				
Konec řešení [ms]	1000	3000	5000	10000	20000	1000	3000	5000	10000	20000
Doba řešení [ms]	1001	3003	5001	10001	20019	1002	3001	5005	10004	20003
Iterace	34	77	127	252	498	25	70	120	225	475
Nalezené cesty	366	1102	1899	3752	7143	361	1054	1792	3324	7032
Stagnace [%]	0	0	0	0	0	0	0	0	0	0
Úspěšné hledání [%]	100	100	100	100	100	100	100	100	100	100
Nejlepší řešení	1604	1544	1472	1453	1429	1537	1490	1475	1421	1408
Průměrná délka všech řešení	5432	3425	5323	4421	4055	4352	5671	4532	4412	4612

Tab 14. Max-min ant system a aplikace lokálního prohledávání

	Bez aplikace lokálního prohledávání					S aplikací lokálního prohledávání				
Konec řešení [ms]	1000	3000	5000	10000	20000	1000	3000	5000	10000	20000
Doba řešení [ms]	1000	3000	5001	10000	20000	1000	3000	5000	10000	20000
Iterace	21	75	124	252	503	28	68	124	241	495
Nalezené cesty	306	1123	1852	3612	7541	361	1011	1842	3671	7412
Stagnace [%]	0	0	0	0	0	0	0	0	0	0
Úspěšné hledání [%]	100	100	100	100	100	100	100	100	100	100
Nejlepší řešení	1588	1520	1479	1448	1423	1590	1486	1482	1455	1416
Průměrná délka všech řešení	6324	6723	6491	6812	6124	7512	7512	7624	6517	7102

Tab 15. Ant colony system a aplikace lokálního prohledávání



Obr. 33 Model prostředí pro aplikaci lokálního prohledávání

6.3 Plánování cesty více robotů

Poslední etapou experimentů bylo ověření funkčnosti plánování cesty více robotů a porovnání sériového a paralelního hledání cesty. Tedy takového hledání cesty, kdy celá kolonie konstruuje řešení pro jednoho robota v čase, nebo kdy je kolonie rozdělena na více podkolonií a každá z nich konstruuje řešení pro jednoho robota.

Plánování cesty více robotů bylo realizováno v modelu prostředí o velikosti 800 x600 bodů s 28 překážkami, viz obr. 34 . Cílem bylo nalézt cestu pro 3 roboty. Startovní a cílové body byly dány následovně:

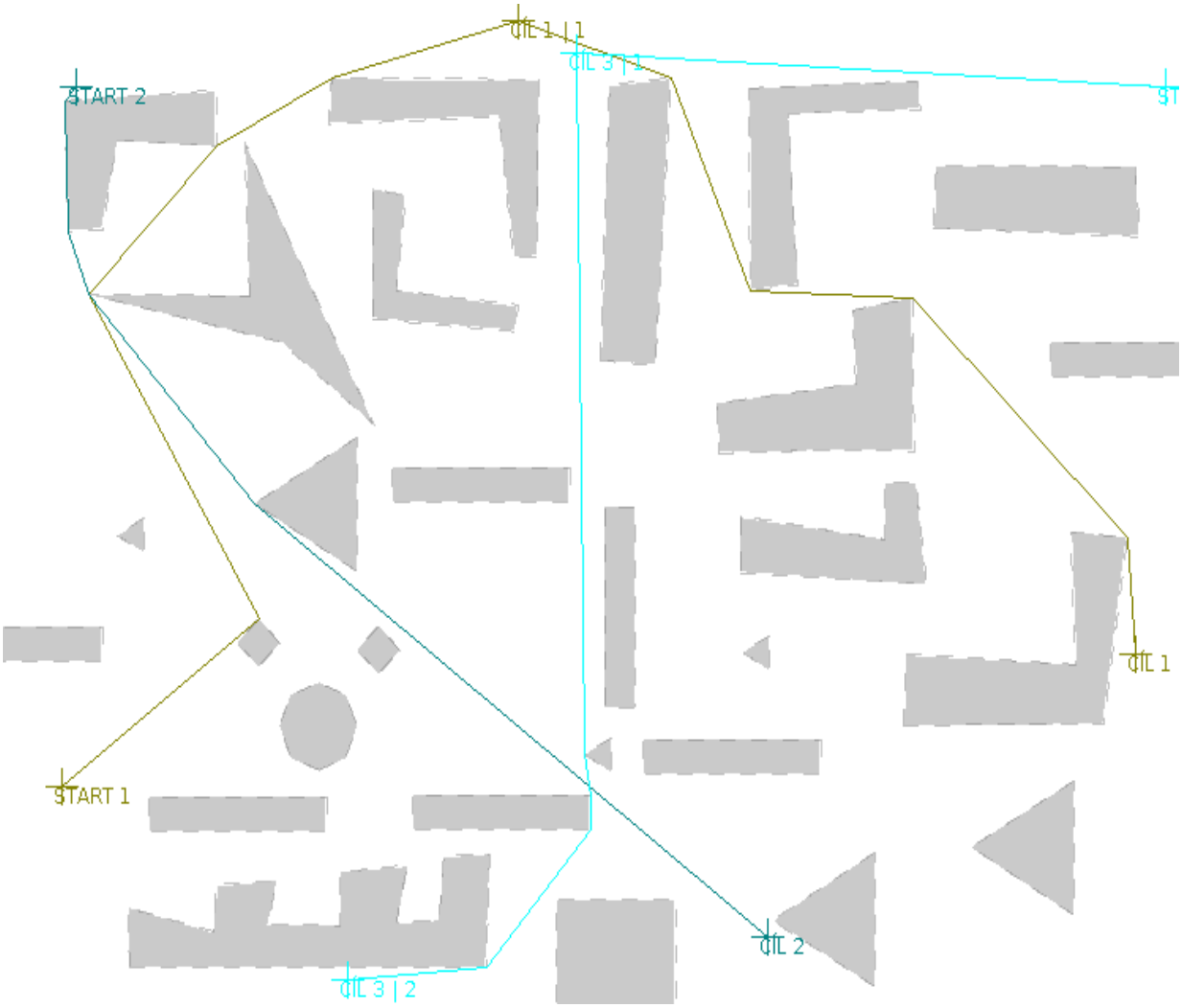
- Robot I: Start (40;470), cíle (350;10) a (770; 390).
- Robot II: Start (50;50), cíl (520;560).
- Robot III: Start (790;50), cíle (390;30) a (235; 585).

Podmínky ukončení hledání cesty byly následující:

- Paralelní hledání: Doba hledání 15 s., počet iterací 3000, 9000 nalezených cest. Stagnace 5% na 400 iterací.
- Sériové hledání: Doba hledání 5 s., počet iterací 1000, 3000 nalezených cest. Stagnace 5% na 400 iterací.

Hledání cesty bylo realizováno všemi výše uvedenými algoritmy. Nastavení parametrů algoritmů je stejné jako v podkapitole 6.1 .

Výsledky hledání cesty všech robotů oběma způsoby jsou dány následující tabulkou:



Obr. 34 Model prostředí pro plánování cesty více robotů

Algoritmus		Ant system		Elitist AS		Rank-based AS		Max-min AS		Ant colony system	
Strategie hledání		S	P	S	P	S	P	S	P	S	P
Nalezené cesty		9020	9015	9014	9020	9023	9012	9031	9020	9035	9012
Iterace		224	212	201	205	201	231	206	212	207	215
Celková doba řešení [ms]		2108	1624	1835	1764	1320	1415	4928	3892	5475	4935
Délka cesty	Robot 1	1439	1398	1275	1275	1302	1371	1365	1315	1335	1372
	Robot 2	742	737	733	733	737	737	733	733	733	742
	Robot 3	1245	1152	1069	1075	1069	1123	1069	1120	1069	1069

Tab 16. *Srovnání sériového a paralelního hledání cesty*

Jak je z tab 16. patrné, paralelní a sériové hledání cesty dosahují v ohledu délky řešení přibližně stejných výsledků. Sériové hledání je však ve většině případů nepatrně rychlejší. Důvod, proč tomu tak je, může být následující: Při sériovém hledání není nutné provádět reininicalizaci celé mravenčí kolonie na nové startovní a cílové body, dochází tedy k úspoře strojového času a tím i k nepatrnému zrychlení celého algoritmu.

7 ZÁVĚR

Cílem této práce bylo popsat problematiku plánování cesty mobilního robotu, popsat principy mravenčích algoritmů, a na základě předchozích dvou bodů navrhnout, implementovat a ověřit metody pro plánování cesty robotu pomocí mravenčích kolonií.

Algoritmy popsané v této práci předpokládají pohyb holonomního robota v daném pracovním prostředí. Prostředí je uvažováno jako dvourozměrné statické s polygonálními překážkami. V tomto prostředí jsou dány startovní a cílové body, přičemž robot musí pro úspěšné splnění úkolu navštívit všechny cílové body. Plánování cesty robota je realizováno skupinou jednoduchých umělých mravenců, kteří spolu stejně jako jejich reálné předlohy nekomunikují přímo, ale změnou prostředí, ve kterém se pohybují.

Algoritmy uváděné v této práci vychází z nejstaršího z mravenčích algoritmů, Ant systemu, a dále z algoritmů rozšiřující původní Ant system. Algoritmy navržené a implementované v této práci prokázaly schopnost řešit problémy plánování cesty i ve složitém prostředí. Tyto algoritmy také byly schopny řešit plánování cesty pro více robotů. Důležitým zjištěním je fakt, že uvedené algoritmy nalezly řešení v každém z provedených experimentů. V rámci experimentů se nepotvrdil původní předpoklad, že rozdíly mezi výkonem algoritmů budou podstatné. Všechny navržené a implementované algoritmy tedy podávají přibližně stejné výsledky.

Dle mého názoru jsou algoritmy popsané v této práci vhodné pro plánování cesty holonomního robota ve statickém prostředí. Pro realizaci plánování cesty neholonomního robota pomocí algoritmů uvedených v této práci by bylo možné uvedené postupy rozšířit způsobem naznačeným v podkapitole 4.8 . Další možné rozšíření algoritmů popsaných v této práci by se mohlo týkat plánování cesty v dynamickém prostředí. Toto rozšíření by si pravděpodobně vyžádalo přepracování modelu prostředí a částečnou úpravu vlastního algoritmu hledání.

SEZNAM POUŽITÉ LITERATURY

[1] GHOLAMI, F.; MAHJOOB, M.J. An Investigation of Parameters in Ant Colony Optimization for a Path Optimization Algorithm. *In Proceedings of the International Conference on Mechatronics and Automation ICMA 2007*, 2007, pp. 463-468.

[2] LIU S.-H.; LIN J.-S.; LIN Z.-S. A Shortest-Path Network Problem Using an Annealed Ant System Algorithm. *In Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, 2005, pp. 245-250.

[3] LIU, S.; MAO, L.; YU, J. Path Planning Based on Ant Colony Algorithm and Distributed Local Navigation for Multi-Robot Systems. *In Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*, 2006, pp. 1733-1738.

[4] VIET, N. H.; VIEN, N.A.; LEE, S. G.; CHUNG, T. Ch. *Obstacle Avoidance Path Planning for Mobile Robot Based on Multi Colony Ant Algorithm. In Proceedings of the First International Conference on Advances in Computer-Human Interaction,* 2008, pp. 285-289.

[5] FAN, X. , et al. Optimal path planning for mobile robots based on intensified ant colony optimization algorithm. *In Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003 IEEE International Conference on* . Volume: 1 , 2003. s. 131 - 136. ISBN 0-7803-7925-X.

[6] GARRO B. A., SOSSA H., VAZQUEZ, R. A. Path Planning Optimization Using Bio-Inspirited Algorithms. *In: Fifth Mexican International Conference on Artificial Intelligence (MICAI'06)*, 2006, pp. 319-330.

[7] LIU, G., et al. The Ant Algorithm for Solving Robot Path Planning Problem. *Information Technology and Applications, 2005. ICITA 2005. Third International Conference on*. Sydney, NSW , 2005. s. 25 - 27. ISBN 0-7695-2316-1.

[8] MA, G.; DUAN, H.; LIU, S. *Improved Ant Colony Algorithm for Global Optimal Trajectory Planning of UAV under Complex Environment*. In International Journal of Computer Science & Applications. Vol. 4 Issue 3. , 2007. s. 57-68. Dostupné z WWW: <http://www.tmrfindia.org/ijcsa/V4I35.pdf>.

[9] TAMBOURATZIS, T. Progressive Optimisation of Organised Colonies of Ants for Robot Navigation: An Inspiration from Nature. *In Adaptive and Natural Computing Algorithms : Lecture Notes in Computer Science,* 2007. s. 649 – 658.

[10] DORIGO, M, STÜTZLE, T. Ant Colony Optimization. : MIT Press, 2004. 305 s. ISBN 0262042193.

[11] MAZZEO, S., LOISEAU, I. An Ant Colony Algorithm for the Capacitated Vehicle Routing. *Electronic Notes in Discrete Mathematics,* Vol. 18, 2004, pp. 181-186.

[12] DORIGO, M.; BIRATTARI, M.; STÜTZLE, T. Ant colony optimization : Artificial Ants as a Computational Intelligence Technique. I*n IEEE Computational Intelligence Magazine. Volume: 1 Issue:4 ,* 2006. s. 28 - 39. ISSN 1556-603X.

[13] VACULÍKOVÁ, M. *Ant Colony Optimization v prostředí Mathematica*. Zlín : Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, 2008. 87 s. Vedoucí diplomové práce Ing. Zuzana Oplatková, Ph.D.

[14] MEC, Martin. *Grafická interaktivní implementace algoritmu Ant Colony Optimization*. Brno:. Masarykova univerzita, 2008. 39 s. Fakulta informatiky. Vedoucí bakalářské práce doc. Ing. Jan

Žižka, Csc.

[15]SEDLÁK, V. *Plánování cesty robota pomocí mravenčích systémů.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2009. 42 s. Vedoucí bakalářské práce RNDr. Jiří Dvořák, Csc..

[16]KOVÁŘÍK, O. *Ant Colony Optimization for Continuous Problems*. Praha, České vysoké učení technické v Praze, Fakulta elektrotechnická, 2006. 69 s.. Vedoucí diplomové práce Ing. Pavel Kordík. Dostupný z WWW: <http://kovarik.felk.cvut.cz/ant-algorithms/>.

[17]ZOTOS, P. *Path Planning with the humanoid robot iCub*. Laussane, École Polytechnique Fédérale de Lausanne, 2009. 53 s. Semestrální práce.

[18] Kyung min Han. *Collision Free Path Planning Algorithm For Robot Navigation*. Columbia, University of Missouri, 2007. 61 s. Diplomová práce. Vedoucí práce Dr. R. W. McLaren.

[19]J. M., Alvarado, et al. Ant Colony Systems Application for Electric Distribution Network Planning. *In Intelligent System Applications to Power Systems, 2009. ISAP '09. 15th International Conference on* , 2009. s. 1 – 6.

[20]NEGENBORN, R. *Robot Localization and Kalman Filters : On finding your position in a noisy world*. Utrecht: Utrecht University, 2003. 156 s. Diplomová práce.

[21]KRČEK, P. *Plánování cesty autonomního lokomočního robotu na základě strojového učení* . Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. 107 s. Vedoucí disertační práce RNDr. Jiří Dvořák, Csc..

[22]BORENSTEIN, J.; EVERETT, H. R.; FENG , L. *Where am I? Sensors and Methods for Mobile Robot Positioning*. University of Michigan. 1996. Dostupné z WWW: <http://www-personal.umich.edu/~johannb/Papers/pos96rep.pdf>.

[23]ŠÍMA, M. *Plánování cesty robota ve spojitém prostředí* . Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2006. 63 s. Vedoucí diplomové práce Ing. Petr Krček.

[24]DLOUHÝ, M.: *Pravděpodobnostní plánování (Robotika > Průvodce)* [online]. 2003, 12. 5. 2003[cit. 2011-5-10]. Dostupné z WWW: <http://robotika.cz/guide/probplan/cs>.

[25]CARPIN, S.; PAVELKO, E. *An experimental study of distributed robot coordination.* In Robotics and Autonomous Systems. Volume 57, Issue 2, 2009. s. Pages 129-133.